VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

VYUŽITÍ GRAFOVÝCH DATABÁZÍ V OBLASTI KYBERNETICKÉ BEZPEČNOSTI
APPLICATION OF GRAPH DATABASES IN CYBERSECURITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Dušan Tichý**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Sedlák**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Dušan Tichý**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
Vedoucí práce:	**Ing. Petr Sedlák**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává bakalářskou práci s názvem:

Využití grafových databází v oblasti kybernetické bezpečnosti

Charakteristika problematiky úkolu:

Úvod
Cíle práce, metody a postupy zpracování
Teoretická východiska práce
Analýza současného stavu
Vlastní návrhy řešení
Závěr
Seznam použité literatury
Přílohy

Cíle, kterých má být dosaženo:

Cílem práce je využití grafových databází v oblasti kybernetické bezpečnosti formou akvizice a srovnání databází, popisem operativních činností bezpečnostních týmů a demonstrací aplikace grafových databází pro potřeby reportingu pro řídící role. Nakonec na modelové situaci srovná ekonomický dopad na systém, který popsanou technologii implementuje a na takový, který podobná preventivní opatření nezavádí.

Základní literární prameny:

ISO/IEC 27035 Information technology Security techniques - Information security incident management. 2016.

KOLOUCH, J. a P. BAŠTA. CyberSecurity. Praha: CZ.NIC, 2019. ISBN 978-80-88168-34-8.

NIST SP 800-61 Computer Security Incident Handling Guide. Gaithersburg: National Institute of Standards and Technology, 2012.

ONDRÁK V., P. SEDLÁK a V. MAZÁLEK. Problematika ISMS v manažerské informatice. Brno: CERM, Akademické nakladatelství, 2013. ISBN 978-80-7204-872-4.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně dne 28.2.2021

L. S.

Mgr. Veronika Novotná, Ph.D. ředitel	doc. Ing. Vojtěch Bartoš, Ph.D. děkan

Abstrakt

Tato bakalářská práce je zaměřena na představení a návrh využití grafových databází v oblasti kybernetické bezpečnosti, konkrétně v oblasti phishingových e-mailů.

Abstract

This bachelor thesis is aimed at introducing and proposing the use of graph databases in the field of cyber security, specifically in the area of phishing emails.

Klíčová slova

Grafová databáze, kybernetická bezpečnost, phishing, e-mail, STIX 2.1., observables, automatizace

Keywords

Graph database, cybersecurity, phishing, email, STIX 2.1., observables, automation

Bibliografická citace

TICHÝ, Dušan. Využití grafových databází v oblasti kybernetické bezpečnosti [online]. Brno, 2021 [cit. 2021-04-29]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/133639. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Petr Sedlák.

Čestné prohlášení

Prohlašuji, že předložená diplomová práce je původní a zpracoval jsem ji samostatně.

Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

Ve Zbýšově dne 29. dubna 2021

……..

(podpis autora)

Poděkování

Tímto bych chtěl poděkovat svému vedoucímu, Ing. Petru Sedlákovi, za jeho vedení, pomoc a podporu při psaní této práce.

Také bych chtěl poděkovat RNDr. Danielu Tovarňákovi Ph.D., svému nadřízenému a oponentovi, za to, že mě uvedl do problematiky grafových databází. Pod jeho vedením jsem získal potřebné znalosti, abych mohl začít pracovat na tomto tématu i v této práci. Zároveň chci poděkovat i svým kolegům z CSIRT-MU, se kterými pracuji na návrhu podobného sytému v praxi. Z tohoto návrhu jsem čerpal mnoho zkušeností pro svou práci.

Nakonec bych chtěl velmi poděkovat svým blízkým za jejich podporu a trpělivost.

OBSAH

ÚVOD

Se strmě rostoucím objemem denně vznikajících informací, které dále zvyšují nároky na kapacitu, dostupnost a propustnost moderních informačních systémů a služeb, roste zároveň také množství kybernetických útoků a jejich sofistikovanost.

Otázka efektivního řízení hrozeb a včasných reakcí se stává stále důležitější. Efektivní řízení hrozeb však potřebuje pro svoji funkci co nejlepší a nejobsáhlejší zdroj informací o bezpečnostních událostech, výsledcích komplexního monitoringu, automatizovaných analýz, zprávách o nových hrozbách přicházejících od třetích stran a od dalších informačních zdrojů.

Pro tyto potřeby je nutné identifikovat jaká data bude třeba sbírat a jaké vazby mezi těmito daty zachycovat. Poté bude nutné zvolit formát, ve kterém budou data reprezentována mezi aplikacemi a při importu / exportu z databáze a který bude zároveň vhodný pro výměnu informací mezi třetími stranami, tedy některý z často využívaných formátů v tomto odvětví.

Dále se budu věnovat kapitole o Observables, ze zmíněného standardizovaného formátu, kterým bude STIX 2.1 a pojmu Observable samotnému.

Nakonec se chci v teoretické části věnovat problematice phishingových e-mailů, jejich významu jako vektoru při napadení systému / uživatele.

CÍL PRÁCE, METODY A POSTUPY ZPRACOVÁNÍ

Hlavními cíli této práce bude představit technologii grafových databází, na příkladech demonstrovat, jak vypadá modelování grafových domén a seznámit čtenáře s možným využitím technologií grafových databází v oblasti informační / kybernetické bezpečnosti, primárně v oblasti detekce phishingových e-mailů s pomocí informací získaných od třetích stran a s řešením situací, které mohou nastat, pokud podvodná zpráva není detekována včas.

V první, teoretické, části seznámím čtenáře s běžně používanou terminologií z oblasti relačních i grafových databází a s modelováním domény v obou přístupech. Dále se v teoretické části budu věnovat pojmu Observable a vybraným částem jazyka STIX 2.1., které pomůžou definovat konkrétní observables. Následně krátce vysvětlím aplikační rozhraní (API), přístupy k tvorbě API: REST a GraphQL. V závěru teoretické části se zaměřím na phishingové / podvodné e-maily.

V druhé části popíšu fiktivní firmu, pro kterou je výhodné popisovaný systém implementovat, a provedu analýzu jejího „současného“ stavu, tedy stavu před implementací. Poté se zaměřím na návrh vlastního řešení systému, který bude čerpat data z firemní infrastruktury a využívat je pro potřeby řešení. Systém bude rozdělen do třech vrstev. První vrstvou je vrstva datových zdrojů. Zde popíšu, z jakých zdrojů by bylo možné data čerpat interně a jaké nástroje nebo služby třetích stran využít pro zkvalitnění detekce, analýzy podezřelých souborů apod. Druhá vrstva je vrstva databázová, týká se tedy hlavně návrhu uzlů a hran, které se budou v grafové databázi vyskytovat. Protože tato práce značně čerpá z jazyka STIX 2.1, budou uzly a hrany vycházet z Observables, tak jak je definuje STIX.

Poslední částí bude ekonomické zhodnocení. V té se zaměřím na cenu návrhu a implementace celého řešení a časového rámce v jakém takové řešení může být vyvinuto.

1 TEORETICKÁ VÝCHODISKA

1.1 Data, informace, relace a znalosti

V práci se budou často opakovat pojmy data, informace a relace. Definice prvních dvou se však v literatuře a mezi odborníky mírně liší, proto první kapitolu věnuji definici častých pojmů tak, aby se pohled čtenáře mohl sjednotit s pohledem mým a pohledem literatury dále použité v této práci.

1.2 Data

V informatice slovo data vyjadřuje údaj nebo hodnotu, která je počítačově zpracovatelná, například bit, obrázek nebo text.

Ve slovníku jsou data definována jako *„informace, konkrétně fakta nebo numerické hodnoty, sjednocené za účelem prozkoumání a pochopení, sloužící k podpoře rozhodování; informace v elektronické podobě...“* (1), z této definice se dá o datech uvažovat jako o informacích, které získávají význam až když jsou zpracovány (zasazeny do kontextu) nějakým dalším subjektem (člověk, algoritmus, UI).

Data budou z pohledu této práce jednotlivé sledované ukazatele, či získané fragmenty z napadených systémů, prakticky se může jednat o konkrétní hodnotu registru, log... Sledovaná data budou definicí velmi podobná Observables.

1.3 Informace

Norbert Wiener definuje informaci jako *„informace je název pro obsah toho, co se vymění s vnějším světem, když se mu přizpůsobujeme a působíme na něj svým přizpůsobováním“* (2)Tato definice, ačkoliv obsáhlá a přesná, není zcela lehká na pochopení a pravděpodobně neposkytne čtenáři jasný obraz toho, co si pod pojmem informace představit.

„Informace jsou data, kterým jejich příjemce přisuzuje určitý význam na základě poznatků, znalostí, vědomostí a zkušeností, kterými disponuje. Snižují u příjemce entropii (neurčitost).“ (3)

Dle těchto definicí lze informace chápat jako data v takovém kontextu, který má pro příjemce význam a ze kterého je schopný získanou informaci pochopit, zpracovat.

Z pohledu informační bezpečnosti budou informace představovat například zpráva s logy (soubor popisující změny v systému) z napadených systémů. Kromě údajů z logů samotných může zpráva obsahovat i popis zdrojového systému (informace z digitální správy aktiv, spuštěné procesy, přihlášení uživatelé), důvod k analýze logu (např. detekce malware na daném systému), detaily šířící se kampaně.

1.4 Relace

Relace v informatice má původ v relační algebře, avšak často je chápána obecněji, i mimo relační databáze, podobně matematické relaci, jako vztah mezi několika prvky či skupinami prvků v konkrétní doméně.

Relace v kontextu práce vyjadřuje spojení celků (dat či informací), typickým příklad je adresa odesilatele a e-mailová zpráva, taková relace by mohla být popsána jako „Autor“ či „Napsal“. Rozpoznání a zachycení důležitých relací je hlavním důvodem pro existenci systému, který budu dále popisovat a který je jedním z hlavních témat práce. (4)

1.5 Znalosti

Znalosti jsou dalším značně abstraktním pojmem, často bývají definovány jako proces aplikace pravidel, na informace. Zahrnovány můžou být také dřívější zkušenosti (například u aplikací umělé inteligence), výsledek přináší širší porozumění než informace samotné.

Příkladem, pokud je detekován a popsán výskyt určitého malware na více systémech, je možné začít nad infekcemi uvažovat jako nad celkem a popsat ho jako kampaň, případně i odhalit různé vzory chování stejného malware, jen v jiných podmínkách. Získávání takových znalostí je základem pro kvalitní řízení hrozeb v organizaci.

(3)

1.6 Relační databáze a RDBMS

RDBMS, v Česku spíše označován jako systém řízení báze dat (relační je většinou vynecháno), označuje soubor několika programů, který tvoří rozhraní mezi databázovým klientem připojující se na systémem poskytované rozhraní a uloženými daty. Poskytuje klientovi řadu funkcí pro řízení a manipulaci se systémem i uloženými daty. (5)

Nejčastější součásti RDBMS popíšu dle softwaru PostgreSQL. Jedná se o jeden z nejpopulárnějších RDBMS, podporuje širokou řadu funkcí, je dobře zdokumentovaný a dostupný zdarma jako opensource software. (6)

1.6.1 Databáze

Databáze je software, který se stará o uložení dat, metadat (popisují strukturu databáze, např. názvy tabulek, sloupců, datové typy, omezení atp.), indexů, procedur a všech dalších dat, které jsou pro práci s databází potřeba. Mělo by platit, že všechna data potřebná k používání databáze se v ní nachází.

Slovo databáze však vyjadřuje jak výše definovaný software, tak i doménu / logické seskupení tabulek se stejným účelem (např. databáze skladu obsahující tabulky zboží, výdej, příjem apod., či účetní databáze), typicky se označením databáze myslí SW, nebo i celý RDBMS (což je sice chybné označení, ale v praxi se často používá). (5) (7)

1.6.2 Tabulka

Tabulka v databázi je základní strukturou pro ukládání dat, je popsána názvem tabulky a jejími sloupci, slouží k popsání jedné entity, tou může být například zákazník, sledované vlastnosti dané entity popisujeme pomocí sloupců tabulky. (5)

1.6.3 Sloupec

Sloupec, nazýván atribut, popisuje jednu vlastnost sledované entity, je definován jako název, datový typ a omezení (např. délky u řetězce). Často používané datové typy jsou následující: (5)

Číselné hodnoty:

- **INT, INTEGER** – celé číslo v rozsahu ($-2^{31}-1$ až 2^{31}), tedy klasický 32bitový INT známý z běžných programovacích jazyků.
- **SMALLINT** – obdoba INT, pouze rozsah omezen na 16bitů.
- **DECIMAL** – velmi přesné desetinné číslo, příklad implementace může být 128bitový INT a 32bitů na znaménko a exponent. V některé literatuře se doporučují proměnné typy (minimalizace chyby zaokrouhlení), ačkoliv moderní přístupy doporučují ukládat všechna čísla, u kterých chyba zaokrouhlení nastat nesmí, jako řetězce.
- **NUMERIC** – obdoba decimal s menší přesností a menšími nároky na paměť, s možností určit počet desetinných míst čísla a počet číslic.
- **FLOAT** – obdoba numeric, nenabízí však možnost určit počet desetinných míst, pouze počet číslic.
- **REAL** – obdoba float, nelze však nastavit žádný parametr, tzv. single preceision.

Vyskytují se i další číselné typy, ale většinou se jedná pouze o modifikace již zmíněných (např. bigint = 64 b INT, protiklad SMALLINT) a podpora záleží na konkrétním databázovém systému.

Řetězcové typy:

- **CHAR** – slouží k uložení řetězce o pevné délce, musí být definován počet bajtů.
- **VARCHAR** – slouží k uložení řetězců o proměnlivé délce, typicky do 255 B, lze definovat minimální a maximální délku, což může pomoct systému při optimalizaci paměti.
- **TEXT** – slouží pro uložení delšího textu, např. komentáře u článku, vyskytuje se často v mnoha variantách (SMALL, MEDIUM,), základní velikost je 64KB.

Datumy a další běžné typy:

- **DATE, DATETIME, TIME** – datové typy pro uložení datumu nebo jeho části, některé implementace podporují i časové zóny.
- **TIMESTAMP** – Časové razítko, počet sekund od začátku Unixové epochy (1.1.1970 00:00:00 UTC). Často využívaný Unixovými operačními systémy.
- **BIT** – většinou uložen jako string (tedy použito 8 bitů pro uložení jednoho)

- **BYTEA** – byte area, pole binárních hodnot

(8) (9)

1.6.4 Integrita a integritní omezení

Integritu dat, potažmo integritní omezení, dělíme do skupin. Každé omezení představuje určité pravidlo, nebo pravidla, která nějakým způsobem ovlivňují tabulky a data v tabulkách. (5) (10)

1.6.4.1 Entitní integrita

Entitní integrita popisuje, že unikátnost každého záznamu (instance entity) je zajištěna pomocí primárního klíče, což je jednoznačný identifikátor konkrétního záznamu v dané tabulce.

Primární klíč může být jeden atribut – typicky používané ID, tedy vzestupné číslování záznamů. Může jej ale také tvořit několik atributů, jenž dohromady tvoří unikátní n-tici, která se v jiné entitě nevyskytuje. Primární klíč může být v každé tabulce pouze jeden.

Primární klíč by měl být minimální, tedy odebráním prvku z n-tice primárního by přestal být unikátním.

Pokud v tabulce existuje více kombinací atributů, které by mohly tvořit primární klíč, označujeme všechny tyto klíče jako kandidátní.

Z kandidátních klíčů vybereme jeden a označíme jej jako klíč primární, ostatní klíče se poté označují jako klíče alternativní. (5) (10)

1.6.4.2 Referenční integrita

Referenční integrita zajišťuje integritu dat mezi tabulkami, tohoto dosahuje tím, že kromě primárního klíče přidává ještě klíč cizí.

Cizí klíč je primární klíč z jedné tabulky, který je atributem tabulky jiné. Hodnota těchto klíčů je shodná, a tak je možné v obou tabulkách identifikovat spojené záznamy. Při návrhu relačních databázích typicky rozlišujeme 3 typy binárních vazeb, tedy vazeb mezi dvěma tabulkami. (5)

1.6.4.3 Stupně vazeb

Krom nejčastěji se vyskytujících binárních vazeb, kterým se budu podrobněji věnovat později, se v databázích vyskytuje i unární vazba, někdy označována jako rekurzivní binární vazba.

Jedná se o speciální případy, kdy záznam v tabulce má vazbu na jiný záznam v té stejné tabulce, například vysokoškolský předmět může mít jako prerekvizitu jiný předmět.

Dalším typem jsou vazby vyšších stupňů než 2, ty se vyskytují méně, jelikož se vyskytují méně i v reálných předlohách, které se snažíme modelovat.

Jako příklad uvedu situaci, ve které by musely platit tyto podmínky:

- Společnost zaměstnává několik pracovníků.
- Každý pracovník může pracovat na několika projektech.
- Každý pracovník na projektu má počítač.
- Počítač musí být přidělen konkrétnímu pracovníkovi na konkrétním projektu.
- (Pokud pracuje jeden pracovník na více projektech, musí mít na každý projekt přidělen notebook jiný.)
- (Žádný notebook nemůže být přidělen více pracovníkům.)

Poslední dvě pravidla jsou redundantní, ale uvádím je pro přehlednost v problému.

Tedy každý technik má vyhrazen jeden konkrétní notebook pro jeden konkrétní projekt a každý notebook je vyhrazen pouze pro konkrétního technika. V takovém případě se jedná o 1:1:1 relaci, kterou nelze vyjádřit spojením několika relací binárních. (10)

1.6.4.4 Vazba 1:1

Označována „one to one“.

Jedná se o nejjednodušší vazbu, pro každý jeden záznam v tabulce A existuje jeden odpovídající záznam v tabulce B, pokud předpokládáme, že cizí klíče nesmí být null (musí obsahovat hodnotu). Příkladem můžou být například tabulka osob a tabulka řidičských průkazů (za předpokladu, že jedna osoba může mít jen jeden platný). Realizována je tak, že tabulky A i B obsahují cizí klíč z opačné tabulky, tedy konkrétní záznam v tabulce A obsahuje jako atribut primární klíč konkrétního záznamu z tabulky B a naopak. (10) (11)

1.6.4.5 Vazba 1:N

Označována „one to many“.

Tato vazba se používá v případě, že pro jeden záznam v jedné tabulce, existuje mnoho záznamů v tabulce jiné. Jedná se o nejčastější vazbu, její užitečnost spočívá hlavně v eliminaci redundantních dat. Typickým příkladem je tabulka zákazník a tabulka objednávka, kde zákazník může mít mnoho objednávek, ale objednávka může patřit pouze jednomu zákazníkovi. Realizuje se tak, že se do tabulky objednávka přidá cizí klíč obsahující primární klíč zákazníka. (10) (11)

1.6.4.6 Vazba M:N

Označována jako „many to many“.

Popisuje případ, kdy jeden záznam z tabulky A odpovídá několika záznamům v tabulce B a zároveň jeden záznam z tabulky B odpovídá několika záznamům z tabulky A. Tato vazba není přímo mezi dvěma tabulkami realizovatelná a je místo toho řešena dekompozicí, vytvoří se spojovací tabulka, na kterou se napojí tabulky A i B vazbou 1:N. Často uváděným příkladem jsou autor a kniha, jeden autor může napsat více knih a jedna kniha může mít více autorů, jedná se tedy o vazbu m:n. Dekompozice takové vazby by se realizovala vytvořením tabulky např. „autor_napsal_knihu“, která by obsahovala cizí klíč z tabulek knih i autorů. (10) (11)

1.6.5 Doménová integrita

Doména definuje všechny hodnoty, kterých můžou atributy nabývat. Doménová integrita pak aplikuje omezení na domény, tedy definuje datový typ, délku, či daný atribut může obsahovat **null** hodnotu, specifikuje blíže dovolené hodnoty (ty můžou být omezeny např. regulárním výrazem nebo jednodušší podmínkou) a také může určovat, zda bude mít atribut výchozí hodnotu, případně jakou. (10) (11)

1.6.6 Relační datový model

„*Model je reprezentace určitého objektu nebo systému z určitého úhlu pohledu, přičemž ostatní charakteristiky se pro jednoduchost zanedbávají“* (5)

V datovém modelu se snažíme zachytit všechny informace, které jsou podstatné pro informační systém, který bude s daty pracovat, např. jméno, příjmení i rodné číslo mohou být důležitějšími údaji než výška, barva vlasů apod.

Relační datový model popisuje všechny entity, jejich významné vlastnosti, tedy vlastnosti, které potřebujeme v databázi, respektive v informačním systému sledovat a vztahy mezi těmito entitami (relace).

Modelování se dělí do dvou vrstev a tří modelů, které na sebe navazují. Modely postupně poskytují vyšší a vyšší úroveň abstrakce. Návrh modelu začíná na nejvyšší vrstvě, modelem konceptuálním. (5) (10)

1.6.7 Konceptuální vrstva

Tato vrstva je výsledkem datové analýzy, jejím cílem je zjistit jaké entity bude model popisovat, jaké jejich vlastnosti budou podstatné a jaké bude naopak vhodné vynechat. Dále je při návrhu třeba ověřit dostupnost všech údajů, které chceme modelem sledovat. Tato fáze je velmi důležitá, protože model je nutné definovat tak, aby odpovídal potřebám jeho uživatelů. Je zde tedy vhodné návrh konzultovat a skutečně ujasnit požadavky na DBMS. (12)

Konceptuální model je většinou realizován graficky jako schéma v jazyce UML nebo jako ER diagram, běžně je abstrakční natolik, že může být implementován ve většině robustních RDBMS. (5) (10)

1.6.8 Implementační vrstva

Implementační vrstvu tvoří model logický a model fyzický.

Logický model reprezentuje databázové schéma v jazyce SQL, jedná se o definici databáze, entit, atributů, klíčů, indexů a omezení. Tento model stále obsahuje míru abstrakce a univerzálnosti, ale jelikož si RDBMS svůj SQL dialekt často drobně upravují, míra abstrakce i přenositelnosti klesá.

Posledním modelem je model fyzický, ten reprezentuje interní databázové schéma, tak jak si jej systém vytvoří z modelu logického. Popisuje způsob uložení dat, případně další specifika konkrétního RDBMS. (5)

1.6.9 ER Diagram

Jelikož budu pro popis modelů v analytické části používat grafické reprezentace, vybral jsem si ER diagram. Ten popisuje vazby mezi entitami, včetně kardinality, omezení a názvu (významu) vazby. Budu používat „inženýrský“ styl zápisu, také známý jako „Crow’s foot“. (11) (13)

1.6.9.1 Entita

Obrázek 1: Entita

(Zdroj: vlastní tvorba)

Entita je v diagramu reprezentována jako čtverec obsahující její jméno. (11)

1.6.9.2 Relace

Relace jsou v ER diagramu reprezentovány jako čáry propojující entity, na kterých mohou být vyznačena kardinalita a modalita. (13)

1.6.9.3 Kardinalita a Modalita

Tyto pojmy vyjadřují omezení aplikované na relaci a zakreslují se jako speciální značky na čáru k entitě, které se týkají.

Kardinalita představuje, kolikrát může být instance jedné entity referencována instancí jiné, relací vázané, entity.

Modalita naopak představuje maximální počet referencí jedné instance entity jinou instancí, relací vázané, entity. (13)

1.6.10 Značení „crows foot“

Značky se umisťují na relační čáru, k entitě, které se týkají, modalita se kreslí blíže k entitě než kardinalita.

Modalita se značí symbolem „vidličky“ (značí několik) nebo paralelní čárou na čáru představující relaci (značí 1).

Kardinalita je vyjádřena kolečkem (nula) nebo paralelní čárou na čáru relace (jedna). (11) (13)

1.6.10.1 Jedna ku jedné

Obrázek 2: vazba jedna ku jedné

(Zdroj: vlastní tvorba)

Nejjednodušší vazba, 1:1, nevyužívá žádné speciální značky.

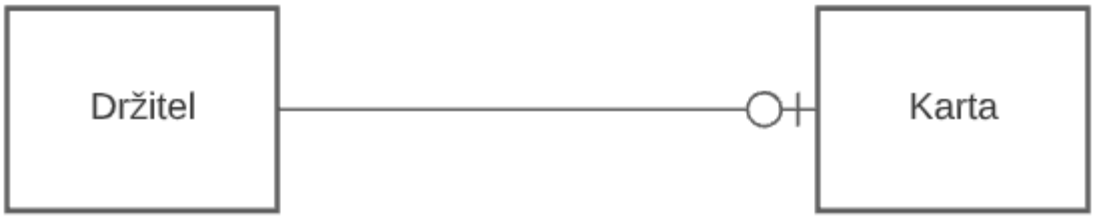
1.6.10.2 Přesně jedna ku jedné

Obrázek 3: přesně jedna ku jedné

(Zdroj: vlastní tvorba)

Vazba 1:1, kde každý držitel musí mít jednu, a právě jednu kartu. Symboly u entity karta vyznačují modalitu 1 a kardinalitu 1.

1.6.10.3 Jedna nebo nic



Obrázek 4: jedna nebo nic

(Zdroj: vlastní tvorba)

Vazba 1:1, kde každý držitel může mít žádnou nebo jednu kartu. Značky vyjadřují kardinalitu nula a modalitu jedna.

1.6.10.4 Jedna ku více

Obrázek 5: jedna ku více

(Zdroj: vlastní tvorba)

Vazba 1:N, kde každý držitel může mít žádnou nebo několik karet.

Obrázek 6: minimálně jedna ku více

(Zdroj: vlastní tvorba)

Vazba 1:N, s integritním omezením, které říká, že firma musí mít jednoho nebo více zaměstnanců. (11)

1.6.11 Normalizace a normální formy

Normalizace je proces sloužící k organizaci dat v databázi, jeho cílem je odstranění / minimalizace redundantních záznamů, při zachování integrity a konzistence.

Při normalizaci se vychází z několika pravidel – normálních forem, každá forma představuje „úroveň“ normalizace databáze, protože na sebe normální formy navazují, tedy pokud chceme databázi normalizovat na určitou normální formu (určitou úroveň), musí být databáze normalizovaná na normu předchozí, tedy musí být aplikována všechna předchozí pravidla.

Většina databází je normalizována aspoň na třetí normální formu. V literatuře se uvádí následující normální formy, pro jednoduchost popíšu princip prvních tří forem.

- UNF či nenormalizovaná databáze – není skutečně normální formou, popisuje systém, který není normalizovaný. Vyskytují se i jiné varianty popisu databáze před 1. NF.
- 1. NF – všechny atributy musí obsahovat atomické (dále nedělitelné) hodnoty. Například atribut adresa může obsahovat mnoho forem zápisu adresy, neúplné údaje apod., po aplikaci 1. NF by se tento atribut měl rozpadnout na atributy: stát, město, ulice, PSČ či ekvivalentní řešení nedovolující složené hodnoty.
- 2. NF – říká, že každý atribut, který není součástí klíče či klíčem, musí být plně závislý na každém kandidátním klíči.
- 3. NF – třetí normální forma se zaměřuje na eliminaci tranzitivních závislostí, ta říká, že žádný atribut, který není primárním klíčem, nesmí být tranzitivně závislý

na žádném klíči. Tranzitivní vazbu budu demonstrovat na následujícím příkladu, jedná se o tabulku popisující radu malého města. (14)

Tabulka 1: Tabulka porušující 3. NF

(Zdroj: vlastní zpracování)

ID	Jméno	Příjmení	Pozice	Mzda
1	Karel	Novák	Starosta	25 000 Kč
2	Lenka	Pravá	Tajemník	22 000 Kč
3	Amy	Pond	Radní	18 000 Kč

Pokud předpokládáme „tabulkové mzdy“, respektive závislosti mzdy na pozici, tak můžeme říct, že atribut „Mzda“ je funkčně závislý na atributu „Pozice“ (mzdu lze určit z pozice). Tím pádem by bylo vhodné přidat tabulku „tabulkových mezd“ a atribut (sloupeček) mzda do ní přesunout. (15) (16)

- Další formy: EKNF, BCNF, 4. NF, ETNF, 5. NF, DKNF, 6. NF

1.6.12 DBMS – Systém řízení báze dat

Systém řízení báze dat je software, který poskytuje jednotné rozhraní ke správě a práci s databází, například umožňuje posílat příkazy v jazyce SQL, které vykoná nad databází. Typicky zahrnuje i správu uživatelů a jejich práv (autentizace a autorizace), na úrovni databází, tabulek i podrobněji, některé minimalistické systémy tyto funkcionality plně neimplementují. Umožňuje současné připojení více klientů či uživatelů. Zabezpečuje integritu dat při selhání tak, aby se systém, pokud možno vždy, nacházel ve stavu, ze kterého je možné jej obnovit. Komplexnější systémy mohou obsahovat i grafické rozhraní, kde je možné systém spravovat i provádět operace s daty (ať už v grafické podobě tabulek, nebo příkazy jazyka SQL). (5) (10)

Typickými příklady jsou jednoduché MySQL a jeho fork MariaDB, populární open-source PostgreSQL, minimalistický SQLite nebo komerčně rozšířený OracleDB. (6)

1.7 Grafové databáze

„*Grafové databáze adresují jeden z největších makroskopických business trendů dnešní doby: využívání komplexních a dynamických vztahů mezi vysoce propojenými daty pro získání znalostí / porozumění a konkurenční výhody.*“ (17)

Z tohoto popisu grafové databáze lze vyvodit většinu důležitých vlastností a rozlišujících faktorů oproti databázím relačním. Grafové databáze se zaměřují na znalosti spočívající v souvislostech a spojitostech mezi daty. Síla takzvaných „connected data“, tedy propojených dat spočívá ve znalostech, které získáme právě pochopením souvislostí mezi jednotlivými prvky, ty mohou tvořit uživatelé, produkty, objednávky a další často popisované entity. (17)

1.7.1 Motivace

Relační databáze, které sice slovo „relace“ obsahují přímo v názvu, vznikaly jako nástroj k digitalizaci dokumentů na papíru, v tabulkách. Ačkoliv relační modely relace zachycují poměrně přesně, v databázi samotné relace jako taková neexistuje,

Existence relace je reprezentována pouze jako atribut nějaké entity a s narůstající propojeností entit narůstá i komplexita při sestavování dotazů a klesá výkon databázového systému.

Navíc čím komplexnější vazby se snažíme v relačním schématu zachytit, tím se i toto schéma často stává komplikovanějším pro inovaci a rozšíření o nové potřeby firmy.

(17)

NoSQL Databáze naráží na podobné výkonové překážky jako databáze relační, hlubší srovnání by už ale zasahovalo spíše mimo rozsah této práce. (17)

1.7.2 Princip

Grafové databáze přistupují k řešení problému s ukládáním vztahů opačným způsobem. Místo toho, aby vazby existovaly pouze na úrovni modelu a při dotazu je databáze musela vytvářet (hledat a spojovat záznamy), jsou data v grafové databázi uložena spolu s vazbami, které mezi nimi existují.

Koncept takzvané „index-free adjacency“, českým překladem by mohlo být „bez-indexové sousedství", spočívá v tom, že na místo referencování spojených záznamů

pomocí primárních, respektive cizích, klíčů, uzly v grafové databázi obsahují přímo referenci na sousední uzly. Díky tomu mohou grafové databáze využívat běžné prohledávání do šířky, které je výrazně levnější a používá se běžně při práci s grafovými datovými strukturami.

Pro nalezení vstupního uzlu, můžeme uvažovat použití B-Stromu pro indexaci uzlů, je třeba vyhledat ID uzlu, nad kterým je dotaz vykonáván, při zvolení vhodné datové struktury pro index to lze také udělat velmi levně. (18) (19)

1.7.3 Rozšíření grafových technologií

Systémy založené na grafech a grafových databázích získávají v posledních letech na popularitě a postupně se rozšiřují z velkých business aplikací typu sociální sítě do menších, běžnějších aplikací. Technologie jako GraphQL, což je dotazovací jazyk pro aplikační rozhraní, který zakládá svou architekturu na doménovém modelování schémat podobným grafům, až po backendy spravující data v grafových databázích např. DGraph a JanusGraph, grafové databáze a navazující architektura se stávají běžnějšími a důležitějšími v moderních aplikacích. GraphQL se umístilo ve statistice State Of JS 2020 v kategorii Technologie, Datová vrstva na prvních příčkách v anketě o spokojenosti, zájmu a povědomí o technologii, na druhé příčce pak ve frekvenci používání. (20) (21) (22)

1.7.4 Graf

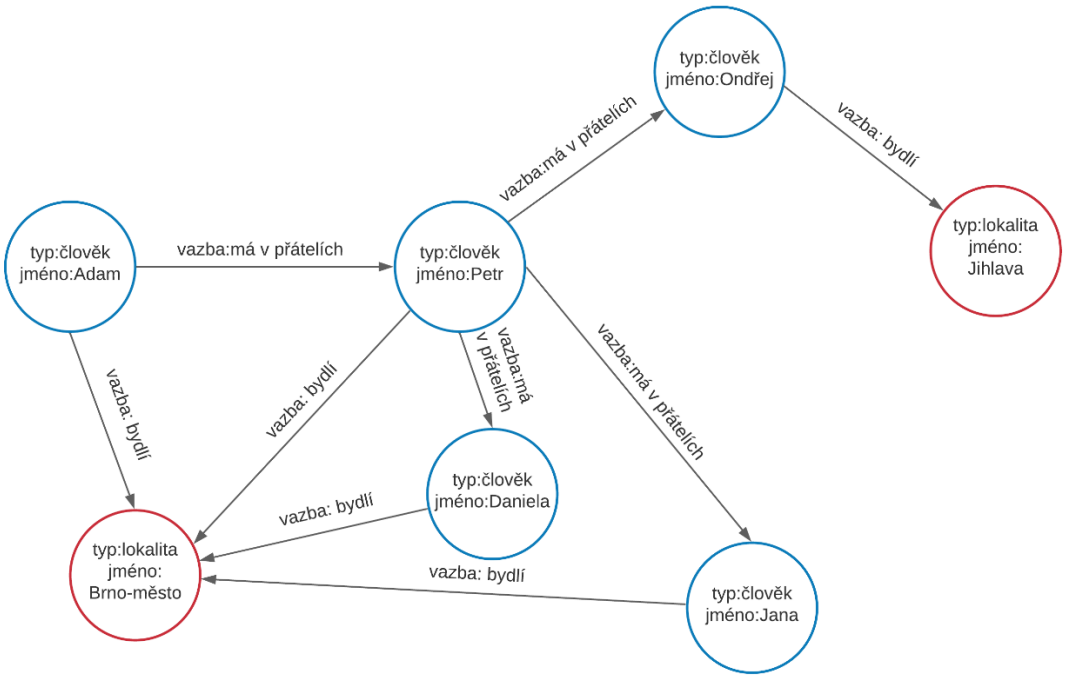
Graf je množina uzlů, případně i hran, které uzly spojují. Hrany lze vnímat podobně jako relace u relačních databází, jsou to jednoduše vztahy mezi uzly. Uzly lze vnímat podobně jako entity relačních databází. Stejně jako databáze relační, i databáze grafové slouží k modelování reálných situací, a ačkoliv je přístup grafových databází v některých ohledech odlišný od databází relačních, mnohé koncepty budou relační databáze připomínat. (17)

1.7.5 RDF Graf

„RDF, Resource Description Framework, je grafový model pro formální definici významu informací.“ Data ukládá do trojic založených na modelu EAV (česky entita, atribut, hodnota), příkladem: „entita: Obchod, atribut:máVNabídce, hodnota:produkt“, entita a hodnota jsou uzly, atribut tvoří hranu. V RDF specifikaci může hodnou být i

skalár, tedy v databázovém kontextu atomická hodnota, některý z podporovaných datových typů jako třeba číslo nebo řetězec. Spojením více EAV trojic pak vzniká RDF Graf. (23)

Jedná se o standardizovaný formát vhodný pro výměnu grafových dat a formalizaci modelů. V rámci zachování intuitivnosti a jednoduchosti budu ale pro vizualizaci používat orientovaný graf, hrany budou představovat vztahy a uzly jednotlivé objekty.



Obrázek 7: sociální graf

(Zdroj: vlastní tvorba)

Tento graf popisuje vztahy mezi lidmi a lokacemi, vazby, které se zde vyskytují jsou „bydlí“ a „má v přátelích“, jedná se o minimalistickou sociální síť, která se snaží doporučovat svým uživatelům další možné přátele, na základě jejich současných přátel a dostupných údajů.

Pokud by automatizovaný systém zmíněné sociální sítě chtěl doporučit např. uživateli Adam nové návrhy na přátelství, mohl by vypadat například takto: „vyber všechny přátele přátel uživatele Adam, kteří bydlí ve stejném městě jako Adam (v Brně)“, výsledkem by byl seznam obsahující uživatele Daniela a Jana, Ondřej by byl vynechán, protože nebydlí ve stejném městě jako Adam.

Jak z demonstrace vyplývá, tento způsob modelování je velmi intuitivní a zachycuje i komplexní vazby, jako přátelé přátel se stejným bydlištěm, jednoduchým a přirozeným způsobem.

1.8 Procházení grafové databáze

V kontextu grafových databází se místo dotazování častěji používá výraz procházení, to více odpovídá činnosti, kterou databáze provádí a často se v informatice používá při popisu prohledávání grafů. Syntaxe jazyka pro procházení grafem se může lišit v závislosti na zvoleném jazyce. Pro svoji práci jsem si vybral již zmíněnou databázi DGraph, která kombinuje populární dotazovací jazyk pro aplikační rozhraní: GraphQL a jeho specifikaci částečně rozšiřuje pod jménem DQL – DGraph Query Language (dříve pod jménem GraphQL+-). (24)

1.9 Observables

„Kyber-bezpečnostní observable je měřitelná / pozorovatelná událost nebo změna měřitelné vlastnosti domény.“ (25)

V reálných aplikacích se jako observables označují data a události, která sbíráme z firemní sítě či lokálnější domény.

Motivací ke členění dat do observables je potřeba standardizace uložení a reprezentace dat, které přicházejí ze sledované domény. Dalším, možná i důležitějším faktorem, který ale bez standardizace dat nemůže jednoduše fungovat, je automatizace řízení hrozeb, respektive snaha automatizovat co nejvíce dílčích procesů, které by jinak musel příslušník bezpečnostního týmu dělat ručně. I částečná automatizace by mohla výrazně snížit nároky na bezpečnostní tým, čím by se snížila potřeba kvalifikace nebo počtu jeho pracovníků. (25) (26) (27)

Observable však není přesně definována jako obecný termín, jelikož její význam se může mírně lišit v závislosti na jejím užití. Pro účely této práce budu brát za příklad definici od MITRE a konkrétní observables založím na observables definovaných ve formátu / specifikaci STIX 2.1.

Užitečnost observables nespočívá jen v jejich schopnosti / vhodnosti pro popis atomických ukazatelů nebo událostí (pomocí observables je například možné popsat artefakty, které popisují vniknutí / kompromitaci nějakého systému, tzv. IOC, „indicator

of compromise“), ale extrémně důležitá a užitečná je jejich schopnost zachycovat i spojení mezi jednotlivými „artefakty“ a tím získat detailní pohled na nějakou událost nebo chování, např. změny chování systému nakaženého malwarem, který se do firemní sítě dostal pomocí phishingového e-mailu. Právě zachycení těchto spojení rozšiřuje možnosti práce se získanými informace, jejich zpracování a vývoje nástrojů, které z těchto dat získávají nové informace, zlepšují kvalitu aktuálních služeb apod. (28) (29) (30)

1.10 STIX 2.1

STIX, Structured Threat Information Expression (volně přeloženo jako strukturovaná výměna informací o hrozbách), představuje jazyk pro výměnu informací o hrozbách. Je vhodný pro serializaci dat. Automatizované systémy, které generují data o hrozbách, můžou do STIXu generovaná data snadno transformovat a ve STIXu je předat dalším systémům, které jsou schopné tento standardizovaný jazyk přijmout. (31)

Hlavní výhody STIXu jsou jeho schopnosti široce popisovat a modelovat domény, zachycovat observables a popisovat vztahy mezi nimi a jejich doménou, dále poskytuje možnost přidávat k datům vlastní metadata, a i rozšiřovat jazyk samotný, například o další observables. Jedná se o jeden z nejpopulárnějších jazyků / formátů používaných v oblasti výměny informací o hrozbách a obecně v oblasti kybernetické bezpečnosti, proto jsem ho zvolil jako formát pro systém, který budu navrhovat v této práci. (31) (32)

1.10.1 Často vyskytující se pojmy ve specifikaci

List – česky seznam, datová struktura podobná poli, na rozdíl od něj není v paměti uložena na jednom místě, jednotlivé bloky jsou spojeny odkazy. Existuje více možností implementace.

Slovník – datová struktura ukládající páry klíč:hodnota, datový typ klíče i hodnoty je omezen implementací datové struktury, každý klíč se může ve slovníku vyskytovat maximálně jednou.

Base64 / b64 – je kódování, které převádí binární data po 8 bitech do textové podoby, respektive do podoby tisknutelných znaků, tedy znaků, které lze reprezentovat jako jeden grafém. Používá 64 znakovou sadu tvořenou číslicemi, velkými i malými písmeny anglické abecedy, + a /. Nakonec = pro zarovnání.

MIME typ – specifikuje kódování řetězce, pokud je to možné měl by odpovídat některému ze standardních typů.

PassiveDNS – služba, která k překladu doménového jména na adresu poskytuje navíc historii překladů, často zahrnuje i výsledky z reputačních databází.

Uvádět nebudu, ale je povinný pro všechny SCO. (31)

SCO – STIX Cyber-observable Object – observable, ve STIXu je definován jako objekt charakterizující hosta nebo síť.

ID – unikátní identifikátor objektu, povinný atribut.

Type – určuje typ STIX objektu, u typu Artifact odpovídá řetězci „artifact“, nemusí se vždy shodovat s názvem SCO. Tento atribut je povinný pro všechny objekty. (31)

TLP – Traffic Light Protocol – široce používaný protokol pro označování citlivosti informací, definuje stavy / barvy TLP:RED, TLP:AMBER, TLP:GREEN, TLP:WHITE, dle stupně závažnosti úniku informace / privátnosti informace. Vychází z dopravního semaforu. (33)

UUID / GUID – Univerzální unikátní identifikátor, řetězec ve tvaru „`123e4567-e89b-12d3-a456-426614174000`“. Jedná se o standardizovanou metodu generování unikátních identifikátorů, s dostatečnou mírou náhodnosti, verze 4, varianta 1, umožňuje vygenerovat 2^{122} unikátních identifikátorů. (34) (35)

1.11 Observables podle specifikace STIX 2.1

V této sekci vyberu pro tuto práci nejdůležitější STIX Observables (SCO's), vybrané observables poté použiji jako základ pro schéma grafové databáze, díky tomu se usnadní transformace a operace s databází přes STIX rozhraní.

1.11.1 Artifact

Artifact je základním STIX Observable objektem a představuje jakoukoliv posloupnost bajtů nebo URL. Posloupnost bytů musí být kódována v Base64, podporuje všechny standardní MIME typy (standardizované identifikátory typu a obsahu média podle RFC 2045). (36)

1.11.2 Directory

Reprezentuje adresář a jeho běžně sledované vlastnosti, konkrétně jsou to cesta (jejíž součástí je i název adresáře), čas vytvoření / modifikace / přístupu a list obsažených souborů a adresářů. (31)

1.11.3 File

Objekt popisuje soubor, běžně sledované vlastnosti jako název (a kódování názvu), velikost (v bajtech), čas vytvoření / modifikace / přístupu. Dále sleduje další vlastnosti jako magic number (hexadecimální číslo, které identifikuje obsah souboru), mime_type a hashe souboru. Součástí objektu jsou také odkazy na nadřazenou složku a obsažené soubory / složky. Soubor samotný je uložen jako Artifact, na který objekt „File“ obsahuje odkaz.

File objekt dále může obsahovat reference na „extension“ objekty, jedná se o rozšiřující objekty, které upřesňují typ souboru. STIX rozlišuje: archiv (jakýkoliv archivační formát), NTFS Soubor (slouží pro zachycení vlastností souborů uložených v souborovém systému NTFS od společnosti Microsoft), PDF, rastrový obrázek a Windows PE Binary (přenositelný spustitelný soubor na operačním systému Windows, patří sem např. portable aplikace). (31)

1.11.4 Domain name

Tento objekt představuje jednoduchý obal pro doménová jména, obsahuje doménové jméno samotné a list překladů. Uvádění překladů je však ve specifikaci uvedeno jako zastaralé. Tuto problematiku je vhodnější řešit například dotazem do PassiveDNS. (31)

1.11.5 Email Address

Obal pro e-mailovou adresu, jeden z velmi běžných observables, zahrnuje adresu, „display name“, což je jméno, které se zobrazí uživateli v e-mailovém klientovi místo adresy (pokud je tato funkce klientem podporována) a identifikátor uživatele, kterému adresa patří. (31)

1.11.6 Email Message

Email message je komplexnější objekt určený pro uložení e-mailové zprávy a extrakci podstatných údajů. Skládá se z následujících položek. (31)

- **is_multipart** – určuje, zda e-mail obsahuje více MIME částí.
- **date** – datum odeslání.
- **content_type** – obsah Content-Type hlavičky, například „text/html“.
- **from_ref** – odkaz na objekt e-mailové adresy, který se nacházel v hlavičce „From“.
- **sender_ref** – odkaz na objekt e-mailové adresy, který se nacházel v hlavičce „Sender“.
- **to_refs** – list obsahující odkazy na e-mailové adresy všech příjemců.
- **cc_refs, bcc_refs** – ekvivalent předchozí položky, ale pro kopie, resp. skryté kopie.
- **message_id** – unikátní identifikátor e-mailové zprávy.
- **subject** – předmět zprávy.
- **received_lines** – list všech „Received“ hlaviček, které se v e-mailové zprávě vyskytují, představují e-mailové servery, přes které zpráva prošla od příjemce po odesilatele.
- **additional_header_fields** – slovník pro další hlavičky
- **body** – tělo e-mailové zprávy, pokud je použito, musí být hodnota položky „is_multipart“ nepravda.
- **raw_email_ref** – odkaz na objekt Artifact, který obsahuje originální „raw“ podobu zprávy.

1.11.7 Email MIME

Je součástí e-mailu, slouží pro popisování komplexních zpráv, jejichž tělo se skládá z několika MIME kódovaných částí, např. obrázek může být součástí zprávy jako samostatný celek. (31)

1.11.8 IPv4 Address a IPv6 Address

Podobně jako u objektu pro doménové jméno, jedná se o obalový objekt, který uchovává IP adresu a její reverzní překlady na DHCP záznamy, domény.

Podobně jako u domén, nedoporučuje se hodnoty překladů uchovávat, jelikož se mění v čase. (31)

1.11.9 MAC Address

Obalový objekt uchovávající MAC adresu, obsahuje validační pravidla pro MAC adresy, STIX konkrétněji požaduje dvojtečky jako oddělovače, malá písmena a vedoucí nuly pro každý oktet. (31)

1.11.10 URL

Obalový objekt pro URL, validačním požadavkem je platná URL podle RFC3986. (31) (37)

1.11.11 Windows Registry Key / Value Object

Objekty pro Klíč a Hodnotu v registru systému Windows. (31)

1.11.12 Process

„*Objekt proces popisuje běžné vlastnosti spuštěné instance programu na nějakém operačním systému.*“ (31)

Jedná se o velmi volně specifikovaný objekt a jeho účelem je v podstatě popsat, jak byl program spuštěn. Definuje tedy následující nepovinné položky:

- **is_hidden** – vyjadřuje, jestli je proces skrytý.
- **pid** – process id – identifikační číslo procesu.
- **cwd** – současná pracovní složka procesu.
- **command_line** – tato položka obsahuje celý příkaz, kterým byl program spuštěn.
- **environment_variables** – slovník proměnných prostředí.
- **openned_connection** – otevřené síťové spojení.
- **creator_user_ref** – specifikuje uživatele který proces spustil.
- **image_ref** – odkaz na spustitelný soubor, který je „obrazem“ spuštěného procesu (program, který byl spuštěn).
- **parent_ref** – odkaz na „rodičovský“ proces, pokud se jedná o proces spuštěný jiným procesem.
- **child_refs** – odkazy na procesy „potomky“ (procesy spuštěné tímto procesem).

1.12 Phishingový e-mail

Phishing, v češtině podvodný e-mail, je speciálním případem nevyžádané pošty, kdy se odesilatel vydává za věrohodnou osobu nebo společnost a snaží se na příjemce působit legitimním dojmem. Po získání důvěry příjemce se od něj snaží získat citlivé údaje, typicky k bankovním, e-mailovým a jiným důležitým účtům, kreditním kartám apod.

Dle statistik se jedná o jeden z nejběžnějších vektorů útoku, 32 % všech úniků dat zahrnovalo phishing jako vektor k získání přístupu. 94 % malware bylo k oběti doručeno e-mailem. (38)

1.13 API: REST a GraphQL

API je zkratka pro aplikační rozhraní, jehož úkolem je zajistit možnost komunikace mezi aplikacemi, propojení různých datových zdrojů a podpora automatizace a integrace.

Aplikační rozhraní definuje, jak je možné s aplikací komunikovat, tedy v jakém formátu očekává požadavek, jaká data musí obsahovat, jakým způsobem probíhá autentizace / autorizace apod. (39) (40)

Rozhraní můžou existovat na různých vrstvách, například aplikační rozhraní operačních systémů je jednou z položek definovaných souborem standardů POSIX. Jeho myšlenkou je zaručit, nebo alespoň usnadnit chod programů na každém operačním systému, který je certifikovaný UNIX, respektive POSIX-compliant (implementuje POSIX, ale není certifikovaný). Většina Linuxových distribucí standard POSIX do značné míry implementuje a existují i POSIX certifikované distribuce. (40) (41)

1.13.1 REST API

REST je akronym pro „REpresentional State Transfer“, jedná se o architekturu pro tvorbu aplikačních rozhraní, která se zaměřuje na výměnu informací mezi distribuovanými (hypermédia) systémy. Informace je v REST architektuře abstrahována jako „zdroj“, jakákoliv pojmenovaná informace může být zdrojem. REST architektura popisuje šest omezení, které musí rozhraní splňovat:

- **Klient-server** – popisuje oddělení odpovědností klienta a serveru. Říká, že klient i server můžou být vyvíjeny nezávisle jeden na druhém, pokud se rozhraní nezmění.

- **Bez stavové** – server žádným způsobem neuchovává relaci nebo stav v jakém je klient mezi jednotlivými požadavky. Stav aplikace (klientské) závisí pouze na klientovi.
- **„Chacheable“** – říká, že zdroje můžou být „cacheovatelné“ (uložené v mezipaměti), takové zdroje musí deklarovat, že můžou být „cachovány“ a mohou být uloženy do mezipaměti serveru i klienta.
- **Uniformní (generalizované) rozhraní** – navrhované rozhraní by mělo mít jasnou strukturu a logický systém, jakým se bude s rozhraním pracovat. Například práce s jedním zdrojem by měla být stejná (či podobná) jako s jiným zdrojem ve stejném rozhraní.
- **Vrstevnatost** – REST umožňuje systém rozdělit do několika vrstev, které nebudou pro klienta z hlediska rozhraní od sebe rozpoznatelné. Například rozhraní na jednom serveru a databáze na několika jiných serverech.
- **Code-on-demand** – (nepovinná položka) – rozhraní může vracet krom statických položek i spustitelný kód.

Ačkoliv REST architektura nespecifikuje HTTP jako protokol, nad kterým má být implementována, ani konkrétní akce pro konkrétní HTTP metody, je s tímto protokolem často spojována. (42) (43)

1.13.2 GraphQL

GraphQL se prezentuje jako dotazovací jazyk pro aplikační rozhraní a zároveň jako serverové prostředí pro vyhodnocování těchto dotazů. Podobně jako REST API jedná se pouze o aplikační rozhraní, které nespecifikuje, co se bude dít na aplikační nebo datové vrstvě (nevyžaduje žádný speciální jazyk / technologii pro aplikaci jejíž rozhraní tvoří, ani neklade požadavky na specifickou databázi). (44)

GraphQL definuje svůj vlastní jazyk pro tvorbu „schémat“, která popisují, jaké objekty, s jakými vlastnostmi rozhraní popisuje (obsahuje). Schémata také slouží k validaci příchozích dotazů na rozhraní. (45)

GraphQL se snaží doménu modelovat jako graf, proto například rozhraní grafové databáze Dgraph vychází z GraphQL, což přináší i nativní kompatibilitu s GraphQL. (24)

2 ANALÝZA PROBLÉMU A SOUČASNÉ SITUACE

Analýzu budu provádět na fiktivní firmě středních rozměrů, která používá standardní řešení v oblasti IT infrastruktury a zaměstnává kyberbezpečností tým.

Před analýzou vymezím její hranice primárně na systémy a služby, které se týkají témata práce a činnosti kyberbezpečnostního týmu, následně se zaměřím na identifikaci pro bezpečnost důležitých aktiv a analýzu jejich zranitelností, primárně na ty, které mohou být monitorovány, případně i automatizovaně. Automatizace se však bude týkat až návrhu řešení. Poté popíšu odpovědné role za konkrétní aktiva.

Následně nastavím stupnici pro klasifikaci aktiv podle dopadu na organizaci při jejich narušení a aktiva klasifikuji. Po klasifikaci identifikuji hrozby a zranitelností daných aktiv, pro zranitelnosti identifikuji důležité bezpečnostní události, které by mohly nastat.

Poté identifikuji pravděpodobné incidenty, které popisují nejpravděpodobnější následky popsaných bezpečnostních událostí.

Nakonec pomocí údajů z dosavadní analýzy navrhnu matici pravděpodobnosti hrozby, jenž pomůže identifikovat rizika, kterým by se měla firma věnovat přednostně. Tato matice umožní sestavit matici úrovně rizik.

Na základě této analýzy určím, na které procesy se zaměřit v procesu automatizace a nasazení systému pro sběr dat založeném na grafové databázi. Této části se budu věnovat v kapitole Vlastní řešení.

2.1 Popis fiktivní firmy

V analýze budu uvažovat firmu středních rozměrů, zhruba 50 zaměstnanců, podniká v oboru vývoje software, většina jejich zaměstnanců jsou vývojáři a designeři, dále management, účetnictví a komunikace se zákazníky.

Analýza bude zaměřena na nejčastěji používané informační systémy a software ve firmě. Zahrnovat bude tedy

- Softwarové repositáře a další pomocný vývojový software – git / gitlab, nexus apod.
- Služby elektronické pošty – zahrnuje procesy související s detekcí a filtrování phishingových e-mailů, snahy propagovat malware či ransomware.

- Další komunikační kanály (jiné než e-mail) - Mattermost, ticketing a další.
- Účetní software a exporty – serverová aplikace běžící na serveru ve firmě a klient na jednotlivých stanicích.
- Externí úložiště – to se dále dělí na hot storage, který bude mít firma na místě a dále cold storage, který bude realizován cloudovou službou.
- Služby pro vzdálený přístup – VPN.

2.2 Analýza situace před implementací řešení

Řešení, kterému se budu ve své práci věnovat, je vhodné hlavně pro velké firmy a situace, ve kterých by komplexita koordinace a náročnost na pracovní sílu bezpečnostních týmů začala představovat exponenciálně narůstající náklad. V takových scénářích začnou skutečně vynikat automatizovaná řešení, často proto, že můžou být snadno škálována, a i když se výkon aplikace zvýší několikanásobně, náklady na údržbu mohou zůstat téměř neměnné, např. výkonnější CPU v serveru nepředstavuje z hlediska údržby téměř žádnou změnu.

V této kapitole se chci věnovat popisu několika scénářů, které se v takových organizacích běžně vyskytují. Popíšu proces, jaký může nastat, jaká jsou běžná řešení a jaké nároky na bezpečnostní tým představuje. V kapitole o vlastním řešení se pak budu věnovat možnostem automatizace procesů, nebo částí procesů, které musí bezpečnostní tým provádět v takových situacích.

Nasazení takového řešení může v praxi i často snížit tlak na bezpečnostní tým, což se může projevit rychlejší dobou odezvy, kvalitnějšími službami, které vyžadují zásah experta nebo nižšími požadavky na proškolenost personálu.

2.3 Problém: šíření phishingového e-mailu na firemní síti

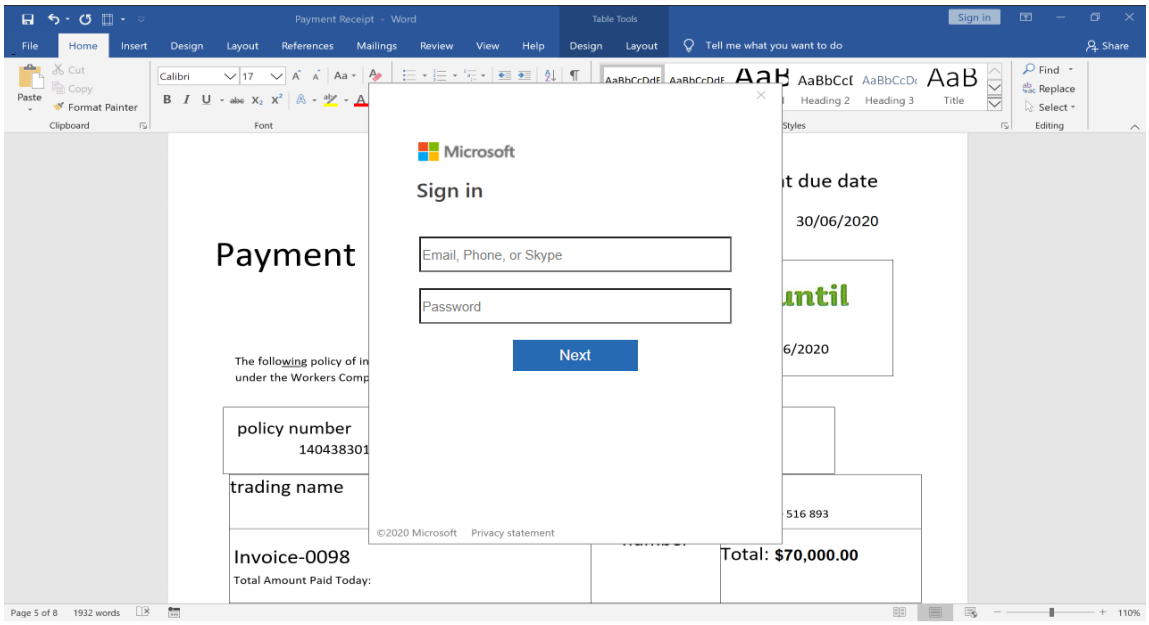
Phishingové e-maily jsou jedna z nejčastějších forem, nejtypičtějším vektorem, útoku na jednotlivce nebo organizace, dokonce až 65 % útočníků používá phishing jako primární vektor útoku. (38)

Ačkoliv se společnosti většinou snaží svůj personál školit o nejběžnějších hrozbách na internetu, kam phishing jednoznačně patří, jedná se stále o relativně úspěšnou formu

útoku. Podvodné zprávy se navíc stávají stále věrohodnějšími, což může pro málo zkušené uživatele představovat značný problém.

Příkladem může být nedávný úspěch phishingové kampaně na žáky a zaměstnance VUT. Rozhodl jsem se založit modelovou situaci na tomto útoku, protože slouží jako vynikající příklad z praxe, který ukazuje že i principiálně jednoduchý útok může být vysoce efektivním, pokud je dobře zacílený.

Podvodný e-mail obsahoval informaci o nezaplacené faktuře a odkaz na textový dokument formátu docx, alespoň zdánlivě. Ve skutečnosti se jednalo o odkaz na závadnou stránku, která byla relativně zdařilou kopií webové aplikace Word Online, zobrazený dokument navíc skutečně obsahoval text připomínající fakturu a přes něj byl zobrazený přihlašovací formulář, více či méně typický pro jednotné přihlášení do služeb od firmy Microsoft. Při vyplnění údajů byl navíc uživatel přesměrován na skutečný soubor s fakturou, veřejně publikovaný na online cloudovém úložišti OneDrive od Microsoftu, které umožňuje dokumenty automaticky otevřít v aplikaci Word Online. Ačkoliv skutečný dokument nepůsobil identicky k dokumentu, který byl na pozadí předchozí stránky s podvodným formulářem, méně pozorný nebo neznalý uživatel mohl snadno uvěřit tomu, že se pouze přihlásil do aplikace OneDrive a nemusel tak ani tušit, že vyplněním svých údajů je odeslal přímo útočníkovi.



Obrázek 8: podvodná stránka

(Zdroj: vlastní tvorba)

Navíc, není výjimkou, že místo toho, aby firma investovala do skutečně kvalitního proškolení každého zaměstnance v oblasti rozpoznání phishingu a nakládání s takovými e-maily, může spoléhat na to, že spamový filtr zajistí dostatečnou úroveň ochrany proti nejběžnějším phishingovým kampaním.

Vhodnost takového přístupu je však diskutabilní, protože v praxi spamový filtr neodhalí všechny podvodné e-maily a příliš striktní nastavení jen zvyšuje risk falešně pozitivních detekcí.

Pokud by firma šla o úroveň výše a snažila se i například analyzovat podvodné e-maily, které spamovým filtrem prošly a byly označeny jako závadné až koncovým uživatelem,

narostly by nároky na personál bezpečnostního týmu několikanásobně. V praxi je ale takové opatření zavádět nutné, protože podvodný e-mail rozpoznaný jedním zaměstnancem nemusí být rozpoznán jiným. Naskýtá se tedy otázka, jak při řešení takových situací postupovat? Častou cestou je manuální trasování podvodného e-mailu v síti organizace a následné informování zasažených uživatelů. Jedná se však o poměrně jednoduchý a repetitivní proces, který je svou podstatou náročný na pozornost pracovníka, a navíc může být značně časově náročný. Dalším faktorem je například to, že nemusí být nutné informovat všechny uživatele, mnohdy se můžeme setkat se situací, kdy zaměstnanec podvodný e-mail odhalí a prostě na něj nezareaguje, nebo jej pouze smaže. Správné vymezení zasažených subjektů snižuje dále administrativní náročnost pro bezpečnostní tým, čím přesněji zacílíme pouze zasažené uživatele, tím méně uživatelů bude tým muset obsloužit. Konečné optimalizace spočívají v používání sjednocených komunikačních kanálů pro stejné kampaně. Případně i v publikování snadného návodu pro mitigaci dopadů, tedy poskytnout obětem například odkaz na změnu hesel, odhlášení všech zařízení, a i specifičtějších instrukcí dle povahy útoku.

3 VLASTNÍ ŘEŠENÍ

Řešení, které budu popisovat, nebude jedna aplikace nebo služba, ale spojení několika modulárních systémů, které zajistí aplikaci potřebnou flexibilitu a jednotlivým komponentám přinese recyklovatelnost (jednu službu by nemělo být problém využít ve více případech / řešeních) a také možnost komponenty jednoduše nahradit.

Začnu popisem problému, na který se systém bude primárně zaměřovat, tedy detekce phishingových zpráv a možných řešení nastalé situace.

Následně se budu věnovat návrhu samotné aplikace, tu rozdělím do tří sekcí, první se bude soustředit na získávání dat z různých datových zdrojů a jejich transformaci do standardizovaného formátů.

Druhá sekce se bude zabývat přijmutím, zpracováním a uložením dat získaných ze služeb z první sekce, tedy samotnou aplikací grafových databází v řešení.

Třetí sekce bude popisovat možnost využití dat uložených v řešení z druhé sekce, navrhnu zde možnosti automatické detekce a reportingu pro bezpečnostní tým a management.

3.1 Detekce phishingových zpráv

Nejlepší způsob jak mitigovat phishingová rizika, je začít kvalitní detekcí, která zachytí většinu běžných hrozeb. Pokud navíc může být detekce obohacena daty z jiných zdrojů, například privátní doménové blacklisty nebo i analýzou obsahu, tak pravděpodobně velmi zredukuje počet „procházejících“ podvodných zpráv.

3.1.1 Detekce

Pro detekci a minimalizaci dopadů phishingových zpráv bude třeba přístup k e-mailovému serveru, na něm můžeme předpokládat běžící spam filtr, který by měl zachytit ty nejběžnější podvodné e-maily a o detekci informovat navrhovaný detekční systém, což může provést například zápisem do logu, pokud nepodporuje notifikace / kód třetích stran, nebo data přímo předat skriptu který se stará o kolekci. Tyto informace se následně zpracují a uloží do databáze, můžeme je poskytnout třetím stranám v rámci výměny informací anebo pomocí nich například změnit úroveň nějakého rizika, které je monitorováno.

Příkladem můžeme dostat informaci o podvodné zprávě šířící se firemními sítěmi, na základě této informace můžeme předpokládat riziko s určitou úrovní, ačkoliv útok na firmu samotnou neprobíhá, jen výskyt zpráv v „okolních“ subjektech může úroveň rizika zvyšovat. Pokud toto riziko vnímáme jako velké, může jej naopak výrazně snížit zpráva, že ho náš spam filtr zvládnul automaticky zablokovat.

Další možností by bylo hrozby s rizikem přesahujícím námi stanovenou úroveň rovnou na spamovém filtru blokovat nebo se pokusit o hlubší detekci, která by mohla vypadat například tak, že každé příchozí zprávy by byly extrahovány observables a porovnány se záznamy v databázi. Tato možnost je už však značně náročná na HW infrastrukturu a výkon systému, prakticky by bylo nejlepší ji aplikovat jen za určitých podmínek, třeba testovat e-maily, které dostaly od spamového filtru nižší reputační ohodnocení, které ale stačí na to, aby prošly filtrem.

Detekce podvodných e-mailů na základě dat získaných od třetích stran je možné provádět i retrospektivně, například každých 24 h v době nízkého využití sítě, například v časy, kdy nejsou zaměstnanci v práci. Podstatným faktorem je také zpětná vazba od uživatelů ve firmě, tou může být nahlášený phishing v e-mailovém klientovi, případě přímé kontaktování bezpečnostního týmu.

3.1.2 Dopad a jeho řešení

V případě, že takový podvodný e-mail prošel do firemní sítě, je nutné riziko eskalovat a přejít z proaktivních opatření na opatření reaktivní, hlavními prioritami je zabránění dalšího šíření a identifikace zasažených uživatelů, případně i škod.

Prvním z kroků, po zablokování zprávy, je nalézt všechny příjemce daného podvodného e-mailu v logu e-mailového serveru a informovat je o tom, že byli cílem útoku, v závislosti na obsahu / podstatě dané zprávy je možné, že uživatelé budou schopni problém vyřešit sami (smazat e-mail) nebo kontaktovat podporu. Taková „úvodní“ zpráva může být do značené míry automatizována, od generování obsahu, či použití šablony, po automatické zaslání všem detekovaným uživatelům. Předpokládám, že firma používá pro interní komunikaci nějaký, pro všechny zaměstnance, standardizovaný komunikační kanál, přes ten bude všem napadeným uživatelům zaslána zpráva s možností odpovědi směřující na pracovníka bezpečnostního týmu.

Zpráva je přitom dále analyzována, ať už automaticky či pracovníkem bezpečnostního týmu. Automatizovaná analýza bude hledat typicky vyskytující se vzory, jako URL adresy, na které musí uživatel kliknout, infikované soubory v přílohách a podobně.

V případě že se útočník bude snažit vynutit klik na odkaz, bude takový odkaz analyzován, pomocí PassiveDNS se zjistí reputace dané domény, uloží se screenshot stránky, kterou měl uživatel navštívit, případně i HTML a související soubory. Dále se z NetFlow analýzy síťového provozu a historie DNS zjistí, kteří uživatelé kliknuli na zkoumané URL, takovým uživatelům bude nejspíše nutné dále věnovat extra péči, například jim sdělit k jakým údajům se mohl útočník dostat, jaká hesla bude třeba změnit apod.

V případě detekce přílohy je vhodné ji analyzovat pomocí anti malware služby, například využít populární VirusTotal, který zkontroluje, zdali se jedná o malware a případně vrátí zjištěné informace. Ty mohou obsahovat i informace o podstatě malware, což usnadní členovi bezpečnostního týmu jeho práci s analýzou a urychlí celý manuální proces.

Manuální analýza může probíhat paralelně nebo následovně po automatické, jejím úkolem je zjistit rozsah škod, tedy proč útok proběhl a jaká data a systémy jsou ohroženy. Po analýze sestaví pracovník zprávu, ve které doporučí manuální kroky nutné k minimalizaci dopadů, například změna hesel apod. Na základě této zprávy pak pracovník podpory vypracuje návod pro uživatele a předá jim ho po stejném komunikačním kanálu, kterým byli uživatelé původně kontaktováni o zjištěné hrozbě.

3.1.3 Manuální alternativa

Automatizovanou detekcí zasažených uživatelů a úvodními skeny je značně zredukován objem repetitivní, monotónní práce, kterou by jinak musel manuálně provádět pracovník bezpečnostního týmu, eliminací takových činností se jednak snižuje časová náročnost v kritických momentech, tedy neprodleně po odhalení hrozby, navíc se snižuje riziko lidské chyby.

Pokud by byly zasaženy stovky uživatelů, bylo by bez automatice takového procesu velmi komplikované všechny uživatele včas kontaktovat a trasovat jejich aktivity na síti. Jistá míra automatizace by se jistě vyskytovala v procesu vždy, ale jen například zpracovat filtrovaný log z DNS může být velmi časově náročné. Síla řešení zde spočívá v propojení

co nejvíce automatizovaných aktivit do jednoho komplexního procesu, který se skládá z více kroků a sám obsahuje prakticky jen jejich spojení, například předání nalezené domény a časového rámce do vyhledávacího dotazu v NetFlow, automatické zpracování takového dotazu a extrakce podstatných dat.

3.2 Architektura systému

Systém rozdělím do tří vrstev, první vrstva bude popisovat datové zdroje, tedy odkud a jak bude systém získávat data, zdroje se budou dále dělit na interní a externí.

Druhá vrstva bude zahrnovat databázi a rozhraní, které bude používáno k interakci s databází.

Třetí vrstva bude popisovat jaké výstupy bude systém nabízet z manažerského hlediska, tedy statistiky o provozu, důležitých událostech a možnost procházet data, dělat nad daty dotazy.

3.2.1 Modulární přístup

Návrh řešení komplexních systému a funkcionalit v IT většinou začíná dekompozicí složitého problému na jednodušší zvladatelné podproblémy. Tento přístup je vývojářům dobře známý a v podstatě bez něj není možné vyřešit složitější problém správně, či vůbec. Tento přístup se v posledních letech promítá více a více i do architektury samotných systémů, tedy namísto budování velkých monolitických aplikací, které v sobě zahrnují širokou škálu funkcionalit, se dnes preferuje rozdělit aplikaci samotnou na několik malých aplikací, tzv. „microservices“, s tím, že každá microservice řeší jeden konkrétní problém, může jím být přihlášení uživatele, získání dat, která se uživateli zobrazí, správa uživatelských profilů apod. Tento přístup přináší i do velkých aplikací jednoduchost a odděluje jednotlivé části natolik, že je možné i jednotlivé aplikace měnit za nové či jiné, bez toho, aby byly ostatní části ovlivněny, tzv. modulární přístup.

Systém, který popisuje tato práce, je navrhován jako plně modulární a předpokládá se, že každý subjekt, který jej bude chtít použít, si upraví nebo přidá vlastní moduly. Může se jednat například o nestandardní funkcionalitu, nebo napojení na již existující systém či aplikaci, kterou má subjekt v provozu. Koncipovat takový systém jako monolitický by bylo z dlouhodobého hlediska velmi náročné na údržbu a potenciálně by přineslo velký technický dluh, protože by bylo velmi těžké zajistit konzistenci hlavních funkcionalit

mezi různě přizpůsobenými nasazeními stejného systému. Modulární přístup dovoluje úpravy hlavních, standardních modulů bez nutnosti brát ohled na moduly, které jsou na ně napojené, samozřejmě za předpokladu, že rozhraní, které tyto hlavní moduly poskytují zůstane stejné, nebo bude pouze rozšířeno.

Díky modulárnímu přístupu bude systém možné distribuovat v podobě, do jisté míry univerzálních, kontejnerizovaných bloků, což usnadňuje automatizaci provozu, konfigurace a škálování systému.

3.2.2 Komunikace mezi moduly

Všechny moduly budou umožňovat komunikaci / ovládání přes standardní rozhraní, podporované rozhraní jsou dnes standardní REST, kvůli jeho rozšířenosti mezi vývojáři a GraphQL, které implementuje možnost vytváření dotazů přímo v požadavku na server a podporuje modelování domén v grafu podobných strukturách, díky tomu je vhodné pro dotazování do grafové databáze.

Preferované API bude GraphQL, protože vývoj a prototypování GraphQL je rychlejší než REST API, dále také modeluje doménu jako graf / grafy, tedy stejně jako je modelován celý systém, z tohoto důvodu nebude modul grafové databáze podporovat jiné dotazování než přes GraphQL, REST funkcionalita by zde přinášela překážky při vývoji a v případě nutnosti ji bude možné doimplementovat nad hotovým schématem. (46)

REST API je podporováno hlavně kvůli zmíněné rozšířenosti a jelikož bude systém spoléhat na velké množství externích služeb a modulů, je výhodné podporovat aplikační rozhraní, které se dnes považuje za oborový standard. Tímto bych nechtěl zpochybňovat kvality REST API, jen si myslím, že pro tento use-case je GraphQL vhodnější alternativou.

3.3 Vrstva 1: Datové zdroje a kolektory

Jak už jsem zmínil, v této sekci se budu soustředit na interní datové zdroje, ze kterých firma bude schopná čerpat data vhodná ke transformaci do standardizovaného formátu, který již bude do jisté míry reprezentovat observables. Dále do datových zdrojů začlením i externí zdroje, jelikož Threat Exchange / Inteligence je jedním z nejlepších způsobů, jak získat informace o aktuálních hrozbách, možnostech jejich detekce a mitigace.

3.3.1 Interní služby

Interní služby jsou nejdůležitější datový zdroj z hlediska detekce anomálií a bez dobře implementovaného monitoringu téměř nemůže systém fungovat. Proto budu věnovat tuto sekci výběru a zdůvodnění konkrétních datových zdrojů a extrahovaných dat.

První problém, který chci zvolenými službami řešit, je phishing. Začínaje detekcí, a to ať na základě předem získaných informací od třetích stran (Threat Exchange) nebo nahlášením od uživatele. Dále zpracování detekovaných e-mailů a budování vlastních dat popisujících podvodné maily a vylepšení aktuálně používaných služeb pro filtraci pošty.

Druhým problémem bude detekce malware na základě získaných informací (také z Threat Exchange) o jeho chování na stanici a síti. Detekce a analýza malware je ale velmi komplexní téma, které je obsahem mimo rozsah této práce i mých zkušeností, popíšu ho velmi zjednodušeně. V rámci své práce chci malware použít k demonstraci možností, které poskytuje komplexní datová základna.

Budeme předpokládat, že od třetí strany dostaneme popsané chování aktuálně šířícího se malwaru, kterým je zasažena i naše síť. Informací, které můžeme dostat je značné množství a detekce je často komplikovaná sama o sobě, omezím tedy problém na několik častých „symptomů“, které lze na infikovaných stanicích nebo síti vypozorovat. V našem případě to jsou: otevřené porty (které nemají být otevřeny běžně, v rámci síťové / firemní politiky), změny v důležitých konfiguračních souborech, běžící procesy, vznik určitých souborů na specifických místech, změny v systémovém registru (systémy MS Windows) a komunikace na určité domény.

Monitorování klientských stanic, tedy souborů, procesů, registrů, portů atd., bude zajištěno běžným monitorovacím software, jako například Nagios, konkrétní řešení záleží na mnoha faktorech a z hlediska řešení by na zvoleném software nemělo záležet, pokud poskytuje potřebné informace, obdobně s monitorováním sítě.

3.3.2 Externí služby

Externích zdrojů existuje mnoho, některé mohou být cíleny na konkrétní sektor (například bankovnictví) nebo plošnější, vyskytují se privátní i veřejné zdroje (placené nebo zcela soukromé), pro příklad jsem vybral program Open Threat Exchange (dále zkráceně OTX) od společnosti AT\&T, s více jako 100 000 účastníky podílejícími se na výměně

informací o hrozbách se jedná o velmi kvalitní a obsáhlý zdroj informací. Poskytuje takzvané IOC (Indicator of compromise), indikátory kompromitace, což jsou forenzní data, která se nacházejí na kompromitovaných systémech a indikují, že systém byl kompromitován - např. napaden malware, IOC jsou velmi blízké observables a mohou popisovat téměř cokoliv systému se týkaje.

Ačkoliv, podobně jako observables, jejich význam spočívá v propojení mnoha jednoduchých indikátorů, které samy problém nedefinují (například změna hodnoty v registru nemusí nutně znamenat výskyt malware, natož přesně určit o jaký malware se jedná), spojení mnoha indikátorů pomůže rozpoznat a popsat „vzorec“ chování závadného software. (47) (48)

Z datových zdrojů jako OTX může firma sbírat relevantní informace, například o zranitelnostech softwaru, který používá, výše popsaném chování malware, phishingu a rozšiřovat si tak detekční schopnosti vlastního řešení.

3.4 Vrstva 2: Databázová aplikace

Úkolem databázové aplikace bude zachytit data z nástrojů popsaných v první sekci, a to jako observables a vazby mezi nimi. Přijímat bude data ve zmíněném formátu STIX 2.1.a transformovat je do grafových uzlů, které budou představovat observables. Neméně důležitým krokem bude transformovat všechny souvislosti a vztahy mezi observables na grafové vztahy, tedy hrany mezi uzly. I když některé grafové databáze dovolují vytvářet nové typy uzlů a hran dynamicky, pro přehlednost, výkon a udržení konzistence dat je vhodné definovat všechny typy uzlů a hran staticky předem. Jednoznačné, dopředu známé, schéma zároveň usnadní vývoj automatizovaných nástrojů.

3.4.1 Návrh datové vrstvy

Návrh datové vrstvy začnu identifikací klíčových observables, které se budou často vyskytovat v procesech popsaných v prvním řešení, zároveň budu modelované uzly grafu zakládat na výše popsaných STIX 2.1. observables. Následně do modelu přidám i související observables, jejichž výskyt může být podmíněn výskytem observables jiných, a nakonec určím, jaké vazby se budou mezi observables moct vyskytnout.

Klíčové observables v případě detekce phishingu budou e-mailová zpráva, adresa, URL a soubor reprezentující přílohu, tyto observables budou v grafové databázi reprezentovány jako následující uzly a hrany.

3.4.2 Hrany

Hrany si lze představit jako ekvivalent relací v relačních databázích, vyjadřují vazby mezi daty, na rozdíl od relačních databází je ale hrana u grafových schémat součástí návrhu jako samostatný „objekt“, který má jméno a další vlastnosti v závislosti na implementaci dané databáze.

3.4.3 Uzel: SCO

SCO, Stix Cyber Obserbale, představuje základní uzel, který definuje společné atributy, všechny další uzly budou z uzlu SCO dědit, tedy obsahovat všechny atributy, co obsahuje SCO.

Atributy SCO:

- **id** – jednoznačný identifikátor, bude jiný od STIXu, jelikož STIXové ID nepřináší žádnou informaci, a ačkoliv pravděpodobnost výskytu dvou UUID / GUID je v praxi téměř nulová, není důvod STIXové ID ukládat ani ho používat jako identifikátor.
- **type** – typ observable, například email_message.
- **specVersion** – verze protokolu STIX ve kterém byla data nahrána (vhodné uchovat z důvodů kompatibility s knihovnami na serailizaci deserializaci ze STIXu)
- **objectMarkingRefs** – reprezentuje označení objektů, například z důvodů důvěryhodnosti, omezení přístupu apod. Založeny na TLP, Traffic Light Protocol. Tyto objekty budou reprezentovány singleton marking uzlem a hranou.
- **granularMarkings** – fungují podobně jako značení objektů, ale umožňují vyšší míru granularity, tedy například označit konkrétní atribut objektu. Reprezentovány podobně jako markings pro celý objekt, hrana bude zahrnovat seznam konkrétních atributů, na které se granulární marking vztahuje.

Hrany SCO:

- **foundIn** – pokud je SCO, nebo observable z SCO vycházející, součástí nějakého jiného objektu, je reference z SCO do nadřazeného objektu realizována touto hranou.

3.4.4 Uzel: Artifact

Reprezentuje STIX artifact, pokud se jedná o text bude uložený v databázi vhodné pro fulltextové vyhledávání, pokud se jedná o URL, bude uložena přímo v grafové databázi.

Atributy Artifactu:

- **mimeType** – stejný význam jako ve výše popsaném STIXu.
- **textRef** – identifikátor textu v databázi pro fulltextové vyhledávání, ve STIXu tomuto atributu odpovídá payload_bin.
- **url** – pokud se jedná o URL, bude zde uložena jako text.
- **hashes**
- **encryptionAlgorithm**
- **decryptionKey**

3.4.5 Uzel: File

Jedná se o zjednodušenou verzi STIX File objektu

Atributy uzlu File:

- **hashes** – list Hashů souboru
- **size** – velikost
- **name** – jméno souboru
- **nameEnc** – kódování jména
- **magicNumberHex** – magic number viz STIX
- **mimeType** – může specifikovat MIME typ souboru, např. „application/msword“
- **ctime** – čas vytvoření
- **mtime** – čas poslední úpravy
- **atime** – čas posledního přístupu

Možné hrany:

- **containsSCO** – obsažené SCO budou propojeny touto hranou.
- **containsArtifact** – spojuje uzel File s Artifact, ve kterém se nachází obsah souboru.
- **parentDirectory** – spojuje File s Directory ve které se nachází.

3.4.6 Uzel: Directory

Jednoduchý obal pro složku, který zachycuje časy vniku, poslední modifikace a přístupu jako samostatné atributy.

- **path** – cesta ke složce.
- **pathEnc** – kódování řetězce který obsahuje
- **ctime** – čas vytvoření
- **mtime** – čas poslední úpravy
- **atime** – čas posledního přístupu

Možné hrany:

- **containsFile** – spojuje uzel se soubory, které obsahuje.
- **containsDirectory** – spojuje uzel se složkami, které obsahuje.

3.4.7 Uzel: URL

Podobně jako ve STIXu jedná se o jednoduchý obal pro URL, který ji umožní reprezentovat jako uzel.

- **value** – obsahuje url.

3.4.8 Uzel: MAC Address

Stejně jako URL jedná se o jednoduchý obalový objekt / uzel.

- **value** – MAC adresa.

3.4.9 Uzly: IPv4 Address a IPv6 Address

Ve STIXu tyto uzly obsahují list překladů na MAC adresy a odkaz na autonomní systém, do kterého patří. Tyto atributy není v grafové databázi nutné uchovávat, jelikož bude tato vlastnost reprezentována hranou.

Atributy IPvX adresy:

- **value** – IPv4 nebo IPv6 adresa.

Možná hrana:

- **resolvesToRefs** – MAC adresy, na které lze IP adresu přeložit v dané síti.

3.4.10 Uzel: Email message

Jedná se o komplexní a rozsáhlý observable, který i ve STIXu obsahuje několik referencí na další observables, které popisují jeho součásti, v grafové databázi se bude dělit na více uzlů a hran. Transformace STIX objektu e-mailové zprávy (a příslušných částí), bude vypadat následovně.

Každý bod reprezentuje atribut STIX objektu email_message.

- **isMultipart** – označuje, jestli se tělo e-mailu skládá z více částí, tento atribut bude uchováván stejně jako ve stixu.
- **date** – datum vytvoření, atribut uzlu.
- **message_id** – hlavička Message-ID, atribut uzlu.
- **subject** – předmět zprávy, atribut uzlu.
- **receivedLines** – identifikace mail serverů, které předávaly zprávu, popsáno výše, reprezentováno jako atribut uzlu.
- **body** – tělo zprávy, uloženo jako odkaz do databáze pro fulltextové vyhledávání. Tento atribut není hranou, protože se jedná pouze o identifikátor do nějaké jiné databáze nebo jiného úložiště, vhodného například pro fulltextové vyhledávání.

Možné hrany:

- **from** – reprezentován hranou mezi objektem e-mailové zprávy a adresy, hrana obsahuje typ závislosti, tedy „from“.
- **sender** – stejné jako from, jen typ závislosti se mění na „sender“.
- **to** – list příjemců, reprezentován mnoha hranami s vazbou „recipient“.
- **cc, bcc** – list subjektů, kterým byla zaslána kopie / skrytá kopie, vazba reprezentována hranou, typ závislosti „cc“, respektive „bbc“.
- **additionalHeaderFields** – další vyskytující se hlavičky, reprezentovány jako samostatný uzel, propojeny hranou.

- **bodyMultipart** – pokud se tělo skládá z více částí, jsou reprezentovány jako MIME objekty a se zprávou spojeny hranou.
- **rawEmail** – STIX artifact, uložený jako artifact uzel, spojený hranou.

Uzel: Port

Tento uzel nevychází přímo ze STIX specifikace, ale představuje přirozené rozšíření. Jeho účelem je spojit nalezený malware s detektovaným otevřeným síťovým portem, což může být jeden z vyskytujících se symptomů na infikovaných stanicích.

Atributy Port objektu:

- **number** – označuje číslo portu, je možné je validovat například na standardní TCP rozsah portů (0)1–65535.

3.4.11 Uzel: MIME type

Pokud se e-mailová zpráva skládá z několika částí, je vhodné je zpracovat samostatně (dovolí to například spojit e-maily podle stejné přílohy).

Atributy MIME type:

- **contentType** – obsah hlavičky „Content-Type“, uložen jako řetězec.
- **contentDisposition** – obsah hlavičky „Content-Disposition“, uložen jako řetězec.

Možná hrana:

- **bodyRef** – odkaz na Artifact nebo File objekt. Oproti STIXu zde chybí atribut „body“, který představuje body uložené přímo jako atribut objektu, jelikož bude dlouhý text, nad kterým má význam dělat fulltextové vyhledávání, ukládán do databáze fulltextové, není důvod ho ukládat i přímo v databázi grafové.

3.4.12 Uzel: Email Header

Tento uzel představuje e-mailovou hlavičku, slouží primárně pro uložení objektů z listu „additional_header_fields“.

Atributy Email Header:

- **key** – klíč, v tomto případě název hlavičky.
- **value** – hodnota, obsah hlavičky.

3.4.13 Uzel: Windows Registry Key

Uzel popisuje klíč registru systémů Microsoft Windows, obsahuje pouze klíč (cesta včetně názvu) a hranu, která jej bude vázat na jeden či více objektů s hodnotou.

Atributy uzlu:

- **key –** klíč, jak je uložený v registru Windows, včetně cesty od kořenového klíče, například. „HKEY_LOCAL_MACHINE\SYSTEM\HardwareConfig\Current“, pouze celé názvy klíčů.
- **modifiedTime –** čas poslední úpravy.

Možné hrany

- **containsValue** – odkaz na objekt „hodnoty“, který je uložen u tohoto klíče.

3.4.14 Uzel: Windows Registry Value

Obalový objekt pro hodnoty uložené v registru systému MS Windows, obsahuje název, obsažená data a kódování.

Atributy uzlu:

- **name –** jméno objektu.
- **data –** řetězec obsahující hodnotu kódovanou dle specifikovaného kódování.
- **dataType –** kódování atributu „data“, musí být jedním z kódování, které podporuje systém Windows pro hodnoty v registru.

3.4.15 Uzel: Proces

Podobně jako ve STIX-u, účelem tohoto uzlu je popsat spuštěný program (jeho instanci).

Skládá se z následujících atributů:

- **isHidden** – boolean, který říká, jestli je proces skrytý.
- **PID** – identifikátor procesu
- **cwd** – aktuální pracovní složka procesu.
- **commandLine** – příkaz, kterým byl proces spuštěn.

Dále od něj mohou vést následující hrany:

- **image** – odkaz na „File“ objekt, který obsahuje soubor, který byl spuštěn pomocí příkazu v atributu „commandLine“.
- **parent** – odkaz na „rodičovský“ proces.
- **child** – odkaz na proces spuštěný tímto procesem (potomek).

3.4.16 Process

„*Objekt proces popisuje běžné vlastnosti spuštěné instance programu na nějakém operačním systému.*“ (31)

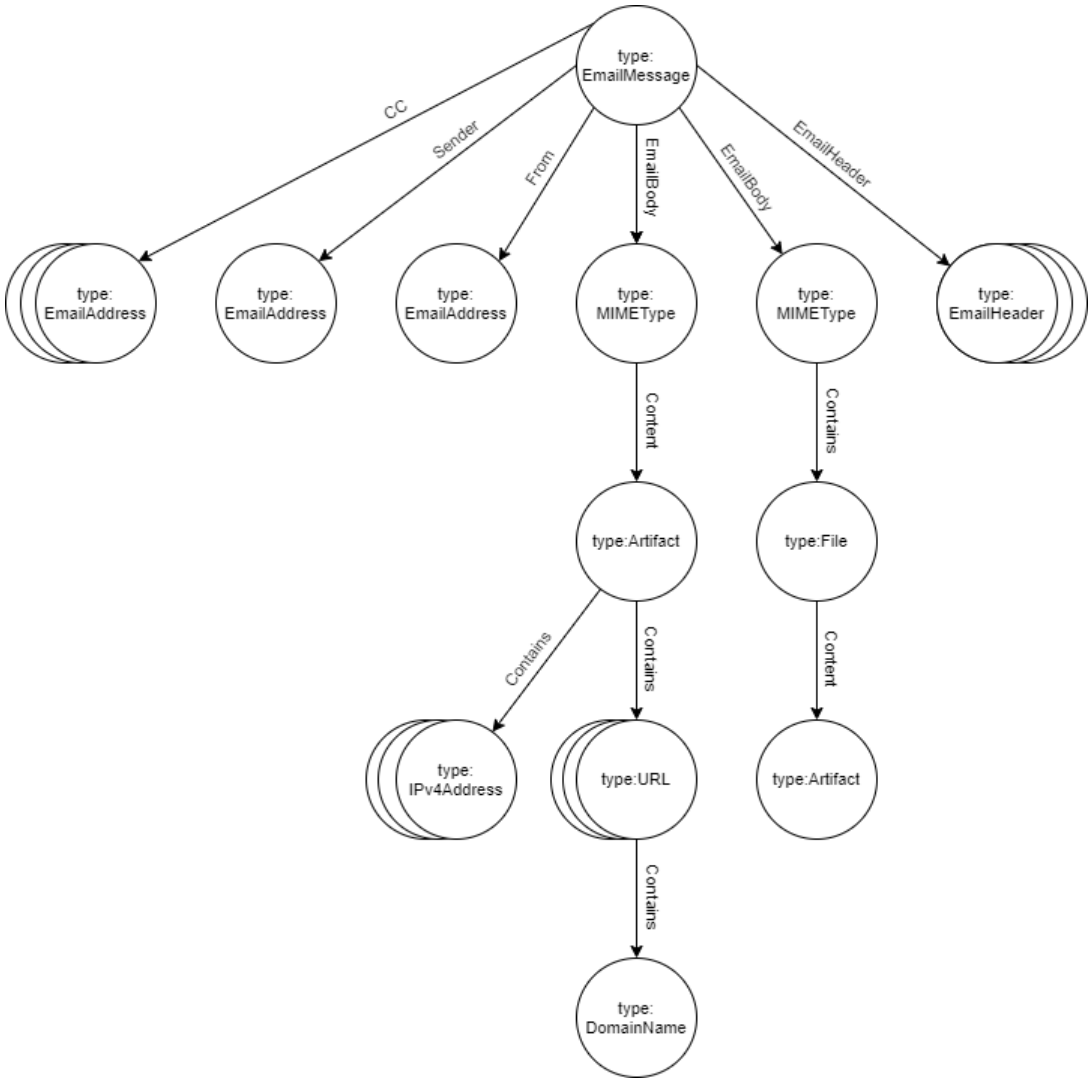
Jedná se o velmi volně specifikovaný objekt a jeho účelem je v podstatě popsat, jak byl program spuštěn. Definuje tedy následující nepovinné položky:

- **is_hidden** – vyjadřuje, jestli je proces skrytý.
- **pid** – process id – identifikační číslo procesu.
- **cwd** – současná pracovní složka procesu.
- **command_line** – tato položka obsahuje celý příkaz, kterým byl program spuštěn.
- **environment_variables** – slovník proměnných prostředí.
- **openned_connection** – otevřené síťové spojení.
- **creator_user_ref** – specifikuje uživatele, který proces spustil.
- **image_ref** – odkaz na spustitelný soubor, který je „obrazem“ spuštěného procesu (program, který byl spuštěn).
- **parent_ref** – odkaz na „rodičovský“ proces, pokud se jedná o proces spuštěný jiným procesem.
- **child_refs** – odkazy na procesy „potomky“ (procesy spuštěné tímto procesem).

3.4.17 Příklady zachycených dat (grafů) v databázi

3.4.17.1 Zachycená e-mailová zpráva obsahující přílohu

Tento graf zachycuje e-mailovou zprávu rozdělenou na konkrétnější observables, které byly ze zprávy extrahovány. Konkrétně se jedná o zpracování obalu a hlaviček, jejich uložení jako dalších uzlů a přiřazení konkrétní relace podle jejich pozice ve zprávě, dále o zpracování všech částí těla e-mailu a příloh, jejichž zpracování se dále liší dle typu obsahu, například z textu jsou extrahovány další, často se vyskytující observables jako URL, potažmo domény a IP adresy. Extrakce těchto dílčích observables bude důležitá pro další zpracování zprávy a získávání znalostí z uložených dat.



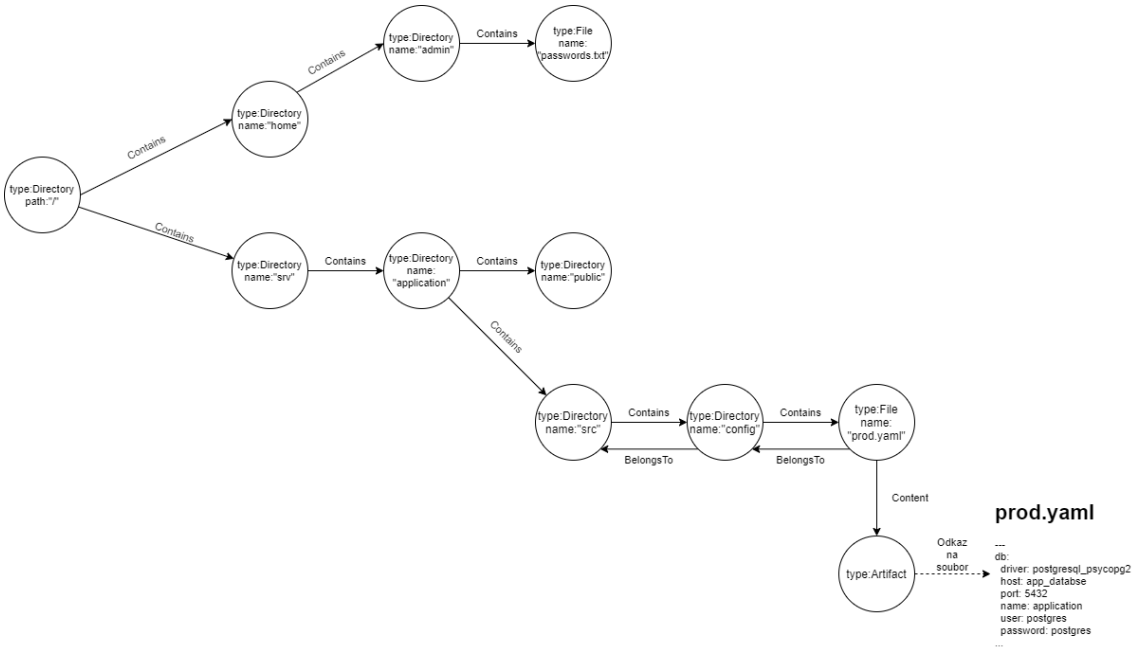
Obrázek 9: E-mailová zpráva reprezentována grafem observables

(Zdroj: vlastní tvorba)

3.4.17.2 Soubory v souborové struktuře systému

Protože phishing je jeden z nejběžnějších kanálů pro šíření malware, lze předpokládat, že bude v databázi vhodné ukládat i data o malware.

Následující graf popisuje důležité informace extrahované z napadeného systému, získané při forenzní analýze, může například reprezentovat soubory, se kterými manipuloval útočník / malware při napadení zmíněného systému.



Obrázek 10: Soubory v souborové struktuře, reprezentovány jako graf z observables
(Zdroj: vlastní tvorba)

3.5 Vrstva 3: Moduly pro reporting

Tato sekce navazuje na Use-case analýzu, která popisuje, jak se uložená data mohou využít v procesu detekce, stejná data mohou sloužit i pro potřeby reportingu pro bezpečnostní tým a management.

Stejně jako každá funkcionalita systému, i tato je závislá na nadstavbě nad databází samotnou, systém bude obsahovat základní reporting modul a samozřejmě možnost přidat další, podle konkrétního nasazení. V této sekci popíšu, co by měl podporovat základní modul a statistky, které by mohly vyplývat z připojených modulů.

Základem modulu bude systém pro autorizaci a autentizaci uživatelů, možnost vytvářet skupiny a role. Díky těmto funkcím bude možné uživatele dělit podle jejich schopností a pozice ve společnosti.

Modul by měl dále umožňovat přístup k datům v grafové databázi i bez toho, aby uživatel musel psát grafové dotazy. Může se jednat o kombinaci jednoduchého „query builderu“, tedy aplikace, která usnadňuje tvorbu dotazů do databáze, může se jednat o technicky náročnější variantu, která umožňuje tvorbu detailnějších dotazů, nebo jednodušší možnost, která je podobná například filtrování v e-shopu.

Kromě tvorby dotazů by modul měl podporovat i dotazy předpřipravené, ze kterých zvládnou číst data i nezaškolení, či méně odborní uživatelé. Tyto dotazy připraví bezpečnostní tým či jiný zkušený uživatel.

Statistiky by mohly rozšířit data plynoucí z připojených modulů, například pro e-mailový modul by bylo vhodné sledovat, kolik se vyskytlo škodlivých zpráv, identifikovaných na základě observables v databázi. Kolik zpráv bylo blokováno spam filtrem z celkového počtu, nejčastější domény a celkový objem pošty v závislosti na čase.

DNS modul by mohl obsahovat podobné statistiky jako e-mailový modul, tedy obecné DNS statistiky, podrobnosti o blokovaných doménách, důvodech blokace apod.

Všechna tato data bude možné exportovat do standardních formátů jako CSV a JSON.

4 EKONOMICKÉ ZHODNOCENÍ

Protože se jedná o relativně rozsáhlý systém složený z mnoha modulů, a navíc závislý na konkrétních potřebách subjektu, může se cena nasazení značně lišit. Díky unikátnosti každého nasazení, se nejedná o řešení, které by bylo běžně komerčně dostupné, alespoň ne v míře, jako je popsáno v této práci.

Řešení je tedy nejvhodnější pro velké subjekty, které si mohou zajistit vývoj značné části systému „in-house“. Cena vývoje spočívá primárně v integraci všech služeb dohromady. Většinu popsaných funkcionalit lze řešit produktem dostupným na trhu, často i open-source, nutné je hlavně software správně vybrat, nasadit a kvalitně integrovat spolu s ostatními službami. Integrace zahrnuje využití platformy pro automatizaci a organizaci procesů jako je Apache Airflow.

Pro menší subjekty by implementace takového systému ve většině případů ekonomicky neměla smysl, jelikož pravděpodobně nebudou mít ani potřebnou infrastrukturu pro sběr potřebných dat a automatické řízení.

4.1 Návrh systému

Za předpokladu, že firma už má potřebnou infrastrukturu a zároveň HW kapacity pro nasazení nového systému, bude hlavním nákladem plánování a vývoj.

Začátkem procesu bude analýza firemních systému, aktiv a funkcionalit, které budou po systému vyžadovány.

Následovat bude identifikace datových zdrojů, jejímž cílem je zjistit, jestli má, v současném stavu, architektura společnosti dostupná všechna požadovaná data. Případně navrhnout jaké rozšíření a modifikace bude třeba provést, pro získání všech požadovaných datových zdrojů.

Po zhodnocení současného stavu bude navrhnut časový rozpočet, ve člověkohodinách, na základě, kterého bude sestaven projektový tým. Předpokládaný tým by se skládal z jednoho projektového manažera, jehož úkolem bude sledování průběhu procesu a komunikace s týmy. Jejich kooperace bude nutná při integraci služeb, například tým infrastruktury, management, který bude používat reporting a další potřebné. Vývoj bude

zajišťovat tým 3-5 vývojářů, s předpokládanou časovou náročností 6 až 18 měsíců, dle náročnosti projektu.

4.1.1 Zhodnocení nákladů analýzy a vývoje

V tomto scénáři uvažuji rozsáhlejší projekt pro 4 vývojáře, analýza bude trvat první měsíc, druhý měsíc začne vývoj, který bude trvat dalších 12 měsíců. Náklady jsou rozpočítány v následující tabulce.

Dále můžeme předpokládat náklady na pracovníky jiných sekcí a konzultace s manažery a dalšími klíčovými pracovníky, zejména z týmů, které budou spolupracovat na integraci. Jelikož se jedná zejména o konzultace, můžeme uvažovat, že za rok nepřesáhnou 500 000 Kč.

Tabulka 2: Rozpočet nákladů na vývoj
(Zdroj: vlastní zpracování)

Role	Počet pracovníků	Mzda	Předpokládané člověkohodiny	Náklady celkem
Projektový manažer	1	500 Kč/h	2080MH	1 040 000 Kč
Vývojář	4	400 Kč/h	7680MH	3 072 000 Kč
Konzultace	-	-	-	500 000 Kč
Celkem	-	-	-	4 612 000 Kč

*MH = člověkohodina

4.1.2 Náklady na dodatečnou infrastrukturu a údržbu

Infastruktura, jak jsem již zmínil, závisí na mnoha faktorech a požadavcích, díky tomu nelze snadno určit jaké náklady by budování dodatečné architektury přineslo, zároveň by ale s největší pravděpodobností sloužila i k dalším účelům.

V závislosti na požadovaných funkcích, služba může běžet na serveru s pořizovací cenou mezi 50 000 Kč a 200 000 Kč.

Největší cenou údržby budou inovace, za předpokladu běžného provozu (bez inovací), by se údržba stala odpovědností bezpečnostního týmu a nepředstavovala by značný náklad.

4.1.3 Návratnost investice

Jelikož se jedná o řešení na míru, které svými funkcemi reflektuje požadavky zákazníka, což znesnadňuje srovnání s komerčními produkty, které se většinou snaží pokrýt širokou oblast problémů.

Komerční SW poskytující alternativní ochranu by představoval endpoint antivirus, běžící na všech klientských stanicích. Jedná se však spíše o komplementární bezpečnostní opatření než o skutečnou alternativu, jelikož zabezpečení koncových bodů je v praxi spíše nutnost, než „možnost“. Systém popisovaný v této práci představuje další vrstvu, kterou útočník bude muset překonat, než se dostane k uživateli.

Nutnost takové vrstvy je vidět také v praxi, kdy i přes všechna existující opatření je phishing stále jedna z nejčastějších a nejúspěšnějších metod útoku.

Cena komerčního řešení, v podobné oblasti, se většinou nachází v rozmezí 10-20 USD za uživatele za měsíc. Za předpokladu, že firma má 500 zaměstnanců, se cena pohybuje v rozmezí 1,3 – 2,6 mil. Kč za rok.

Předpokládejme, že firma zakupuje licenci ke komerčnímu SW na 5 let, získá tím zvýhodněný tarif 12 USD za uživatele (kupuje 500 licencí). Platit bude jednorázově při nákupu.

Celková cena takové investice by byla 360 tis. USD, při aktuálním kurzu 21,5 Kč / USD, zhruba 7 750 tis. Kč. Systém popisovaný v této práci představuje tedy 3 150 tis. Kč úsporu po pěti letech. I za předpokladu dalšího rozvoje systému, a tedy extra nákladů, bude vývoj vlastního systému představovat značnou úsporu.

ZÁVĚR

V této práci jsem seznámil čtenáře se základy relačních a grafových databází a způsobem, jakým jsou domény modelovány pro implementaci v odpovídajících databázích.

Poté jsem se věnoval pojmu Observable, jeho významu v informační bezpečnosti a sběru dat do grafových databází. Na tuto část jsem navázal seznámením s jazykem STIX 2.1., konkrétně s jeho částí o SCO (STIX Cyber-Observable). SCO, které budou mít význam v praktické části, jsem dále popsal tak, jak je zachycuje specifikace.

Dále jsem stručně uvedl phishingové e-maily, jelikož se nezabývám přímo klasifikací a analýzou z oblasti sociálního inženýrství, ale místo toho jsem se chtěl zaměřit na jejich význam z pohledu observables, detekce a výměny informací o hrozbách, většinu problematiky jsem přesunul do sekce s vlastním řešením.

Konec teorie jsem věnoval aplikačním rozhraním, konkrétně populárním REST API a GraphQL.

V analýze současného stavu a situace jsem navrhl fiktivní firmu, pro kterou by bylo výhodné implementovat systém založený na grafové databázi, jehož účelem by bylo vylepšit, tedy zdokonalit bezpečnost v oblasti šíření phishingových e-mailů.

Praktickou část jsem začal praktickými problémy s detekcí phishingu a možnostmi vylepšení na základě dat z různých zdrojů, následně jsem také popsal možné dopady a jaké vylepšení by bylo možné provést v oblasti „disaster recovery“.

Na základě těchto problémů / řešení jsem začal popis navrhovaného systému, vysvětlil jsem jeho modularitu a komunikaci mezi těmito moduly a dělení modulů do vrstev.

Poté jsem se věnoval návrhu jednotlivých vrstev, začínaje „sběrači“ dat z interních a externích zdrojů a přístupem k propojeným službám. Další vrstva se zabývá uložením sbíraných dat ve standardizovaném formátu a návrhem datových struktur pro grafovou databázi: SCO uzly a hrany. Popsané SCO jsem doplnil příklady, které demonstrují observables a vazby mezi nimi, uložené v grafové databázi, respektive v grafu.

V poslední části jsem zvážil, jaké náklady by pro vybranou firmu přinesl vývoj a implementace popsaného systému a případně nutnou provozní infrastrukturu.

Praktické aplikace grafových databází zatím nejsou tak rozšířené jako klasických, relačních. I samotná oblast kybernetické bezpečnosti je stále prudce rostoucím oborem s velkým prostorem pro nové technologie. Myslím si, že pro grafové databáze existuje ještě mnoho využití, stejně jako existuje spousta způsobů, jak naložit s daty, která do nich budou sbírána.

CITOVANÁ LITERATURA

(1) Data. *Cambridge Dictionary* [online]. [cit. 2021-04-28]. Dostupné z: https://dictionary.cambridge.org/dictionary/english/data?q=data

(2) *Informace, komunikace a myšlení: úvod do informační vědy*. Druhé, přepracované vydání. Praha: Karolinum, 2005. ISBN ISBN 978-80-246-1037-5.

(3) KOZUBEK, Libor. *MANAGEMENT ZNALOSTÍ*. Ostrava, 2012. ISBN ISBN 978-80-248-2583-0. Učební text. VŠB – Technická univerzita Ostrav.

(4) A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*. 1970, (387), 11.

(5) *Databázové systémy*. První. Hradec Králové: Gaudeamus, 2012. ISBN ISBN 978-80-7435-203-4.

(6) Stack Overflow Developer Survey 2020. *Stack Overflow* [online]. [cit. 2021-04-28]. Dostupné z: https://insights.stackoverflow.com/survey/2020#most-popular-technologies

(7) What a Relational Database Is. *Oracle* [online]. [cit. 2021-04-28]. Dostupné z: https://www.oracle.com/database/what-is-a-relational-database/

(8) *PostgreSQL Data Types* [online]. [cit. 2021-04-28]. Dostupné z: https://www.postgresql.org/docs/10/datatype.html

(9) *FrontBase SQL 92 Datatypes* [online]. [cit. 2021-04-28]. Dostupné z: http://www.frontbase.com/docs/5.3.1.html

(10) *DATABASE MODELING AND DESIGN: Logical Design*. Fifth Edition. Burlington: Elsevier, 2011. ISBN ISBN 978-0-12-382020-4.

(11) *Datové a funkční modelování*. 1. vydání. Brno: Akademické nakladatelství CERM, 2006. ISBN ISBN 80-214-3252-7.

(12) *Principy datového modelování* [online]. [cit. 2021-04-28]. Dostupné z: https://krokodata.vse.cz/DM/Principy

(13) *Crow's Foot Notation* [online]. [cit. 2021-04-28]. Dostupné z: http://www2.cs.uregina.ca/~bernatja/crowsfoot.html

(14) *Database Third Normal Form Explained in Simple English* [online]. [cit. 2021-04-28]. Dostupné z: https://www.essentialsql.com/get-ready-to-learn-sql-11-database-third-normal-form-explained-in-simple-english/

(15) *Description of the database normalization basics* [online]. [cit. 2021-04-28]. Dostupné z: https://docs.microsoft.com/cs-cz/office/troubleshoot/access/database-normalization-description

(16) *Normální formy* [online]. [cit. 2021-04-28]. Dostupné z: http://lucie.zolta.cz/index.php/iformacni-systemy-databaze/50-normalni-formy

(17) ROBINSON, Ian, Jim WEBBER a Emil EIFREM. *Graph Databases*. 2nd Edition. Sebastopol: O'Reilly Media, Inc, 2015. ISBN 978-1-491-93200-1.

(18) VILHENA, Thomas. *Index-free adjacency* [online]. [cit. 2021-04-28]. Dostupné z: https://thomasvilhena.com/2019/08/index-free-adjacency

(19) JAIN, Manish. *Dgraph: Synchronously Replicated, Transactional and Distributed Graph Database*. , 11.

(20) *How Facebook matured its data structure and stepped into the graph world* [online]. [cit. 2021-04-28]. Dostupné z: https://neo4j.com/news/how-facebook-matured-its-data-structure-and-stepped-into-the-graph-world/

(21) *Thinking in Graphs* [online]. [cit. 2021-04-28]. Dostupné z: https://graphql.org/learn/thinking-in-graphs/

(22) *State of JS 2020* [online]. [cit. 2021-04-28]. Dostupné z: https://2020.stateofjs.com/en-US/

(23) *Data Modeling with RDF(S)* [online]. [cit. 2021-04-28]. Dostupné z: https://graphdb.ontotext.com/free/devhub/rdfs.html

(24) *DQL Query Language* [online]. [cit. 2021-04-28]. Dostupné z: https://dgraph.io/docs/query-language/

(25) BARNUM, Sean. *A (very) Brief Introduction to the Cyber Observables eXpression (CybOX)* [online]. In: . s. 43 [cit. 2021-04-28]. Dostupné z: https://cybox.mitre.org/documents/Cyber%20Observable%20eXpression%20(CybOX)%20Use%20Cases%20-%20(ITSAC%202011)%20-%20Sean%20Barnum.pdf

(26) *Overview of Cyber Observables (CyboX) and the Importance of Information Security Policies for Compliance* [online]. [cit. 2021-04-28]. Dostupné z: https://flank.org/faqs/what-is-cyber-observables-cybox

(27) *What is CybOX? How do you use a CybOX object?* [online]. [cit. 2021-04-28]. Dostupné z: https://cyware.com/educational-guides/cyber-threat-intelligence/what-is-cybox-how-do-you-use-a-cybox-object-af90

(28) *Indicators of compromise* [online]. [cit. 2021-04-28]. Dostupné z: https://docs.servicenow.com/bundle/quebec-security-management/page/product/threat-intelligence/concept/c_IoCs.html

(29) *What are STIX and TAXII* [online]. [cit. 2021-04-28]. Dostupné z: https://cyware.com/educational-guides/cyber-threat-intelligence/what-are-stix-and-taxii-beee

(30) *Malicious E-mail Indicator With Attachment* [online]. [cit. 2021-04-28]. Dostupné z: https://stixproject.github.io/documentation/idioms/malicious-email-attachment/

(31) *STIX™ Version 2.1* [online]. [cit. 2021-04-28]. Dostupné z: https://docs.oasis-open.org/cti/stix/v2.1/cs01/stix-v2.1-cs01.html

(32) RAMSDAE, Andrew, Stavros SHIAELES a Nicholas KOLOKOTRONIS. *A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages*. , 22. Dostupné z: doi:10.3390/electronics9050824

(33) *Traffic Light Protocol* [online]. [cit. 2021-04-28]. Dostupné z: https://www.cisa.gov/tlp

(34) *A Universally Unique IDentifier (UUID) URN Namespace* [online]. [cit. 2021-04-28]. Dostupné z: https://tools.ietf.org/html/rfc4122

(35) Universally unique identifier. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001 [cit. 2021-04-28]. Dostupné z: https://en.wikipedia.org/wiki/Universally_unique_identifier

(36) *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies* [online]. [cit. 2021-04-28]. Dostupné z: https://tools.ietf.org/html/rfc2045

(37) *Uniform Resource Identifier (URI): Generic Syntax* [online]. [cit. 2021-04-28]. Dostupné z: https://tools.ietf.org/html/rfc3986

(38) *Phishing Facts* [online]. [cit. 2021-04-28]. Dostupné z: https://www.phishingbox.com/resources/phishing-facts

(39) *What is an API?* [online]. [cit. 2021-04-28]. Dostupné z: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

(40) *Is Linux POSIX Compliant* [online]. [cit. 2021-04-28]. Dostupné z: https://linuxhint.com/is_linux_posix_compliant/

(41) *POSIX Standard* [online]. [cit. 2021-04-28]. Dostupné z: https://linuxhint.com/posix-standard/

(42) *What is REST* [online]. [cit. 2021-04-28]. Dostupné z: https://restfulapi.net/

(43) *REST Architectural Constraints* [online]. [cit. 2021-04-28]. Dostupné z: https://restfulapi.net/rest-architectural-constraints/

(44) *Introduction to GraphQL* [online]. [cit. 2021-04-28]. Dostupné z: https://graphql.org/learn/

(45) *What is GraphQL?* [online]. [cit. 2021-04-28]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-graphql

SEZNAM OBRÁZKŮ

SEZNAM TABULEK